НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК 004.031.43

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»
зі спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Система управління комплексом підтримки життєдіяльності рослин»

Виконав:
студент VI курсу, групи ІТ-93МП
Суліменко Микита Олександрович

_____

Керівник:
доцент каф. АУТС, к.т.н., с.н.с.
Кравець Петро Іванович

_____

Рецензент:
професор каф. АСОІУ, д.т.н.,
Жаріков Едуард В'ячеславович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський)
Спеціальність – 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Суліменко Микиті Олександровичу

1. Тема дисертації «Система управління комплексом підтримки життєдіяльності рослин», науковий керівник дисертації Кравець Петро Іванович, к.т.н., доцент, затверджені наказом по університету від «26» 10 2020 р. №3132-с

2. Термін подання студентом дисертації 18.12.2020

3. Об’єкт дослідження: управління життєвими процесами рослин

4. Перелік завдань, які потрібно розробити: Аналіз проблем автоматизації вирощування рослин, огляд існуючих рішень, розробка системи керування комплексом підтримки життєдіяльності рослин

5. Орієнтовний перелік графічного (ілюстративного) матеріалу: Діаграма прецедентів, Структурна схема системи, Діаграма станів процесу_запиту і отримки даних, Схема взаємодії компонентів у режимі реального часу, Схема авторизації у системі, Діаграма розгортки програмного продукту, Схема бази даних, Діаграма компонентів клієнтської частини.

6. Дата видачі завдання 02.09.2020

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	02.09.2020	
2	Опрацювання літературних джерел	03.09.2020 – 17.09.2020	
3	Аналіз предметної області	18.09.2020 – 25.09.2020	
4	Проектування системи	26.09.2020 - 17.10.2020	
5	Розробка системи	18.10.2020 - 08.11.2020	
6	Розробка стартап-проекту	09.11.2020 – 16.11.2020	
7	Оформлення дисертації	17.11.2020 – 17.12.2020	
8	Захист дисертації	23.12.2020	

Студент Микита СУЛІМЕНКО

Науковий керівник Петро КРАВЕЦЬ

АНОТАЦІЯ

Магістерська дисертація на здобуття ступеня магістру на тему “ Система управління комплексом підтримки життєдіяльності рослин”: 100с., 24 рис., 27 табл., 8 додатків, 27 джерел.

Об'єкт дослідження – управління життєвими процесами рослин.

Мета роботи – розробка системи, що забезпечує управління апаратним комплексом моніторингу та підтримки життєдіяльності рослин.

У магістерській дисертації розглянуто питання автоматизації керування процесами підтримки життєдіяльності рослин. Описано архітектуру системи для реалізації веб-додатку, проведено її декомпозицію на складові модулі та обрано найбільш оптимальні технології розробки. У роботі вказано на конкурентні переваги та проведено мануальне тестування. Проведено роботу по аналізу ринкових можливостей запуску стартап-проекту програмного засобу, враховано всі ризики, маркетингові стратегії та ключові переваги концепції потенційного продукту. Було побудовано бізнес-план подальшого розвитку системи.

Ключові слова: ВИРОЩУВАННЯ РОСЛИН, СИСТЕМА КЕРУВАННЯ АПАРАТНИМ КОМПЛЕКСОМ, ВЕБ-ДОДАТОК, АРХІТЕКТУРНА ДЕКОМПОЗИЦІЯ

ABSTRACT

Master's thesis for master's degree on the topic "Control system for plant life support complex": 100p, 24 figures, 27 tables, 8 annexes, 27 sources.

Object of study – control of plants` vital processes.

The purpose of the work – development of a system that provides control of a hardware complex for monitoring and maintaining plant life.

The master's thesis deals with the issue of automation of plant life support processes control. The study describes the system architecture for implementing a web application, its decomposition into composite modules, and the most optimal development technologies. The paper indicates the competitive advantages and reveals manual testing. Special attention is given to analysis of the market opportunities for launching a software startup project, taking into account all the risks, marketing strategies and key advantages of the potential product concept. The study includes a business plan for further development of the system.

Key words: PLANT GROWTH, HARDWARE COMPLEX CONTROL SYSTEM, WEB-APPLICATION, APPLICATION ARCHITECTURE

ЗМІСТ

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

AJAX - Asynchronous Javascript and XML

API – Application Programming Interface

CSS – Cascading Style Sheets

DOM – Document Object Model

EDI – Electronic Data Interchange

FTP – File Transfer Protocol

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IoT - Internet of Things

JS – JavaScript

JSON - JavaScript Object Notation

JWT – JSON Web Token

NPM – Node Package Manager

ODM – Object Document Mapper

OSI – Open Systems Interconnection

REST – Representational State Transfer

SOLID – single responsibility, open-closed, Liskov substitution, interface segregation, dependency inversion

SOA – Service-oriented Architecture

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

UML – Unified Modeling Language

URL – Uniform Resource Locator

W3C – World Wide Web Consortium

WSDL – Web Service Definition Language

XML – eXtensible Markup Language

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

СУБД – система управління базами даних

ВСТУП

На сьогодні IoT (Internet of Things) є невід'ємною частиною людського життя. Це концепція обчислювальної мережі об'єктів, що можуть як взаємодіяти між собою, так і з навколишнім середовищем, допомагаючи людині вирішувати різноманітні проблеми. Однією з таких проблем є обмеженість ресурсів та необхідних знань у сфері виробництва аграрної продукції. На даний момент не існує доступної, зручної у керуванні системи, яка дозволяла б ретельно налаштовувати процеси розвитку рослини, враховуючи усі необхідні фактори.

Актуальність роботи полягає в тому, що наразі у світі існує проблема зміни клімату, через яку і окремі невеликі домогосподарства, і великомасштабні виробництва повинні оптимізувати свої виробничі процеси, щоб за умов обмеженості природніх ресурсів більш економічно та з меншою екологічною шкодою розвивати та підтримувати виробництво, максимізуючи розміри врожаю. Щоб задовольняти ці вимоги, виникає необхідність ретельного налаштування параметрів, які впливають на розвиток рослин, відслідковування зовнішніх факторів, реакції рослин при певних сценаріях, та швидкому редагуванні умов підтримки життєдіяльності рослин.

Метою є розробка системи, що забезпечує управління апаратним комплексом моніторингу та підтримки життєдіяльності рослин.

Об'єкт дослідження: управління життєвими процесами рослин.

Предмет дослідження: система управління комплексом підтримки життєдіяльності рослин.

Завдання роботи:

– аналіз сценаріїв життєвого циклу рослини;

– огляд існуючих рішень та порівняння з системою, що розробляється;

– розробка системи по управлінню апаратним комплексом моніторингу та підтримки життєдіяльності рослин;

– проведення експериментального дослідження;

– розробка стартап-проекту;

– аналіз отриманих результатів та створення висновків.

Методологія базується на дослідженнях у роботах «Agricultural IoT and Decision Support for Precision Smart Farming», Annamaria Castrignano; «Smart Plant Monitoring System Using IoT Technology», Ankur Kohli.

Матеріалом для дослідження були сучасні системи «розумного дому», системи автоматизації теплиць, гідропонік та «гроубоксів».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні положення

Прогнозується, що протягом цього століття глобальні температури зростуть на 1,5–5,9 ° C, і ця зміна, ймовірно, вплине на середню кількість опадів, передбачаючи, що дефіцит води, можливо, буде найсерйознішою загрозою для стійкого сільського господарства. У цьому відношенні інвазивні бур’яни, маючи ознаки, які краще пристосовані до стресового стану посухи, ніж сільськогосподарські культури, додають занепокоєння щодо стійкості врожаю. Підраховано, що загальна чисельність населення світу може досягти 9,15 млрд. У 2050 р. [1] , а для збільшення світового виробництва продовольства потрібно робити ще більший прогрес у сільському господарстві в галузі врожайності сільськогосподарських культур та в практиках, які більш відповідають природі.

В останні десятиліття в сільському господарстві стався ряд технологічних перетворень. Завдяки різним "розумним" сільськогосподарським системам автоматизації процесів життєдіяльності фермери отримали повний контроль над процесом вирощування худоби та виробництва сільськогосподарських культур. Зараз ці процеси більш передбачувані – це відчутно підвищує ефективність та економічний прибуток агробізнесу.

Прогноз темпів розвитку рослин в сільськогосподарській галузі має в практичному сенсі не меншу, а часто навіть більшу цінність для конкретного користувача-агронома. Користувач має оперувати різноманітними даними про стан екосистеми, в якій знаходиться рослина, передбачувати їхні зміни та оптимізувати процеси, які впливають на життєдіяльність рослин. Загальновідомо, що більшість параметрів біологічної сфери – характеристик рослини – не є константами. Вони динамічно змінюються по ходу сезону вегетації. І для більшості цих параметрів визначає їх величину є не фізичний (календарний) час, а час біологічний (вік рослини), і на передній план виходить завдання знаходження відповідності між фізичним і біологічним часом, та визначення змін у сценарії їхнього життєвого циклу.

Таким чином, блок опису розвитку становить центральну частину будь-якої, навіть самої примітивної, системи виробничого процесу.

На допомогу приходять автономні системи контролю життєдіяльності рослин, завдяки яким можна абстрагуватися від календарного часу (пори року) і зробити акцент лише на біологічному віці рослини.

Навколишня середа є одним з найважливіших факторів успіху процесу зрощення рослини. Фактори навколишньої середи можна розділити на три групи:

– абіотичні;

– біотичні;

– антропогенні.

Чинники описано на рисунку 1.1.

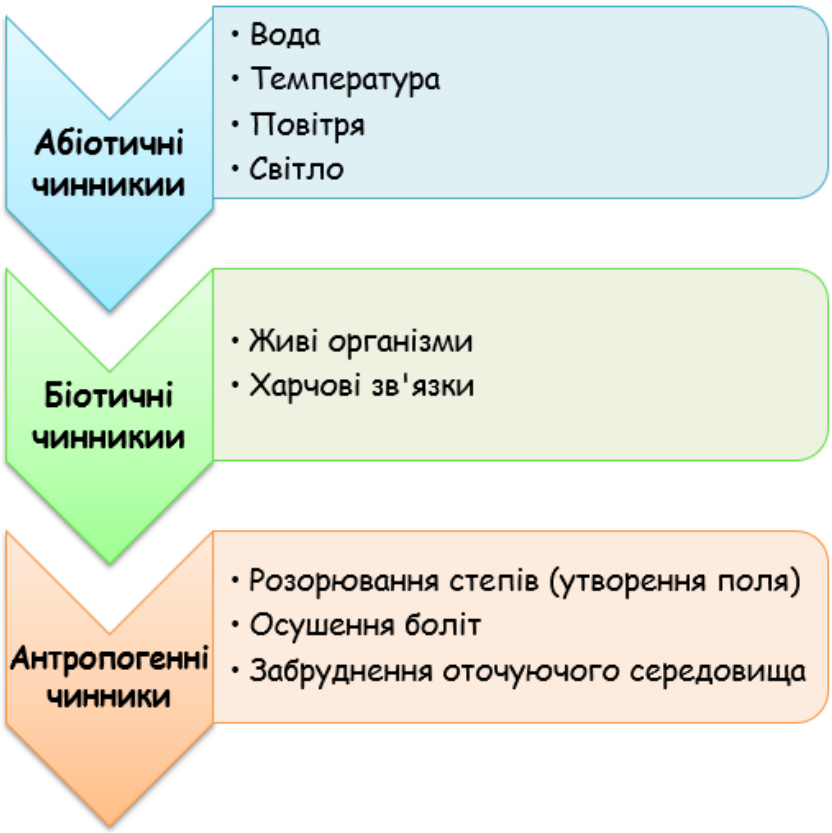

Рисунок 1.1 – Фактори навколишнього світу

Останні складаються зі змін, які були внесені у природу людиною. Це включає різні види засмічення, наприклад, радіоактивні відходи. Біотичні фактори

зумовлюють взаємодію між різними представниками у популяції. Абіотичні фактори є факторами неживої природи, такими як сонячне світло, вода, клімат та інше. Саме абіотичні фактори будуть розглянуті при обчисленні необхідних даних та конструюванні програмного забезпечення.

1.2 Змістовний опис

Зрощення є важливим процесом у будь-якому сільському господарстві, але «автоматизація» з'явилася в зрошувальному виробництві раніше, ніж у сільському господарстві. Рослинам потрібна вода, задовільний рівень кислотності землі, рівень CO2 та достатнє освітлення. Різницею між двома агро-сферами стало розуміння того, що існує певний час, коли їм це потрібно.

Враховуючи критерії розвитку для кожної з рослин, за допомогою автоматичних комплексів контролю життєдіяльності рослини, система керування може аналізувати різні життєво важливі фактори розвитку рослини та оптимізувати графіки втручання у їхню життєдіяльність.

Розглянемо фактори які безпосередньо впливають на рослину:

– вологість ґрунту;

– рівень кислотності ґрунту;

– рівень засоленості;

– вмісту азоту, фосфору, калію;

– освітлення;

– рівень O2 та CO2;

– вегітаційні індекси (NDVI).

Фактори розвитку, які безпесередньо впливають на рослину, показані на рисунку 1.2.

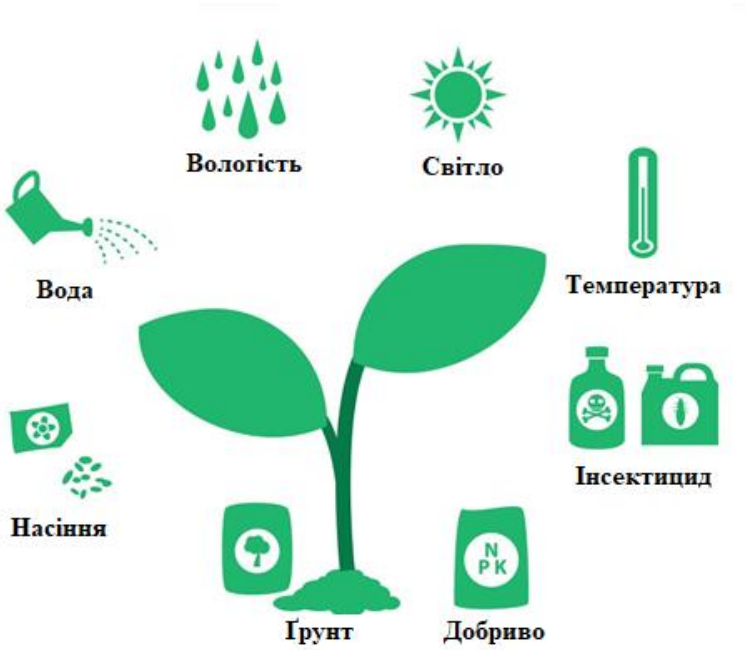

Рисунок 1.2 – Фактори розвитку рослини

Далі буде розглянуто деякі з факторів, які впливають на рослину, більш детально. Вологість ґрунту визначає вміст води у ґрунті. Цей показник впливає на багато факторів. Сенсор показано на рисунку 1.3

Рисунок 1.3 – Сенсор для вимірювання вологості ґрунту

Рослини здійснюють фотосинтез і накопичують суху речовину. А величина накопичення безпосередньо відображається на динамічних змінах висоти рослини, товщини стебла, площі листя та врожайності. У момент, коли кількість вологи недостатня, спостерігається погане зростання, зменшення площ фотосинтезу листків та зменшення врожаю. Лист – основне місце фотосинтезу. Ріст листя дуже чутливі до водних умов. Швидкість фотосинтезу тісно пов'язана з водним станом рослин. Експерименти показали, що коли вміст води в рослинних тканинах буде близьким до насичення, фотосинтез буде найвищим, коли вмісту води буде занадто багато, продихи будуть пасивно закриватися, а фотосинтез буде загальмований. Коли рослинам не вистачає води, фотосинтез зменшується, коли листя в'януть через сильну нестачу води, фотосинтез різко знижується або навіть припиняється.

Підтримання листя у вертикальному положенні відбувається за умови оптимального тургорного тиску у рослині. Явище в'янення рослини, яке виникає, коли рослині не вистачає води, є проявом зниженого тургорного тиску. Коріння рослин - основні органи, що поглинають воду. На його розвиток впливає багато аспектів, але головним є вологість та аерація ґрунту. Розподіл вологи в ґрунті впливає на вертикальний розподіл кореневої системи. Коли вміст води у ґрунті високий, опір ґрунту до дифузії коренів стає меншим, що сприяє появі нових коренів. Ґрунт зазвичай містить певну кількість доступної води, тому сама коренева система не схильна до дефіциту води. Коли ґрунт сухий або запас води недостатній, коренева система поглинає обмежену кількість води і, по-перше, задовольняє власні потреби.

Таким чином, до надземної частини рослини надходить мало води. Тому, коли вологість ґрунту недостатня, вплив на надземну частину рослини більший, ніж вплив на підземну частину. Навпаки, якщо вологості ґрунту занадто багато, а умови аерації ґрунту погані, вплив на підземну частину більший, ніж надземну. Як результат, відношення коренеплодів до пагонів зменшиться. Помірний і повільний дефіцит води може збільшити абсолютну вагу кореня, пригнітити ріст надземних частин, зменшити накопичення сухої речовини надземних частин і зменшити урожай. Але дослідження показали, що певний період дефіциту води сприяє поліпшенню врожайності та якості. Посуха на ранній стадії може підвищити посухостійкість рослини на пізній стадії. А

м’яка посухостійкість на стадії розсади може посилити “компенсаційний ріст” кореневої системи та покращити посухостійкість рослини. Поглинання води є головною умовою проростання насіння. Тільки після того, як насіння поглинуть достатньо води, поступово розпочнуться різні фізіологічні та біохімічні ефекти, пов’язані з проростанням. Це пов’язано з тим, що вода може набрякати і пом’якшувати насіннєву оболонку, тоді кисень може легко проникати через насіннєву оболонку в насіння для посилення дихання зародка. У той же час ембріону легко пробитися через насіннєву оболонку.

Існує багато досліджень, щодо впливання вологості ґрунту на результати зрощування. В них досліджено вплив підвищення вмісту вологи в ґрунті на температуру ґрунту, відбивальну здатність ґрунту та накопичення теплоти ґрунту[2]. Результати показують, що збільшення вологості зменшує різницю температур ґрунту між денним та нічним часом, що забезпечує захист кореневої системи рослин від різких та різких змін температури ґрунту. Також виявлено, що поглинання сонячної енергії зростає із збільшенням вмісту вологи, що призводить до більшої ємності зберігання тепла при більш високому вмісті вологи. Нарешті, швидкість росту та врожайність рослини зросли за рахунок зміни клімату рослин при більш високому вмісті вологи.

Датчик вологості ґрунту вимірює відсоток вологості шляхом, що відповідає існуючим міжнародним стандартам вимірювання вологості ґрунту. Він призначений для моніторингу вологості ґрунту, наукових експериментів, розумного зрошення, теплиць, квітів та овочів. Більшість датчиків вологості ґрунту призначені для оцінки об’ємного вмісту води в ґрунті на основі діелектричної проникності, об’ємної проникності ґрунту, ґрунту. Діелектричну проникність можна розглядати як здатність ґрунту передавати електрику. Діелектрична проникність ґрунту зростає із збільшенням вмісту води в ґрунті. Така реакція зумовлена тим, що діелектрична проникність води набагато більша за інші компоненти ґрунту, включаючи повітря. Таким чином, вимірювання діелектричної проникності дає передбачувану оцінку вмісту води.

Рівень кислотності ґрунту дає різні підказки про властивості ґрунту і легко визначається. Найбільш точним методом визначення pH ґрунту є pH-метр. Другий спосіб, який є простим і легким, але менш точним, ніж використання pH-метра, полягає у використанні певних індикаторів або барвників. pH-метр показано на рисунку 1.4.

Рисунок 1.4 – pH-метр

Вплив pH на ґрунт достатньо великий, він визначає розчинність мінералів або поживних речовин. Чотирнадцять із сімнадцяти необхідних поживних речовин рослини отримують із ґрунту. Перед тим, як поживна речовина може бути використана рослинами, її потрібно розчинити в ґрунтовому розчині. Більшість мінералів та поживних речовин більш розчинні або доступні в кислих ґрунтах, ніж у нейтральних або слаболужних. Фосфор ніколи легко не розчиняється в ґрунті, але

найбільш доступний у ґрунті з діапазоном pH, що центрируется близько 6,5. Надзвичайно і сильнокислі ґрунти (pH 4,0-5,0) можуть мати високі концентрації розчинних алюмінію, заліза та марганцю, які можуть бути токсичними для росту деяких рослин. Діапазон pH приблизно від
6 до 7 сприяє найбільш вдалому отриманню поживних речовин. PH ґрунту може також впливати на ріст рослин, впливаючи на активність корисних мікроорганізмів. Бактерії, що розкладають органічні речовини ґрунту, перешкоджають сильним кислим грунтам.

Це запобігає розпаду органічної речовини, що призводить до накопичення органічної речовини та зв’язування поживних речовин, особливо азоту, які містяться в органічній речовині. Багато досліджень кажуть про те, що цей показник ґрунта є чиненайважливішим фактором успішного взрощування рослини[3].

У міру того, як ґрунти стають більш солоними, рослини не можуть витягувати потрібну кількість води з ґрунту. Це пояснюється тим, що коріння рослин містять різну концентрацію іонів (солей), які створюють природний потік води з ґрунту в коріння рослин. Оскільки рівень солоності в ґрунті наближається до рівня коріння, вода стає все рідше потрапляти в корінь. Насправді, коли рівень солоності ґрунту досить високий, вода в коренях витягується назад у ґрунт[4]. Рослини стають нездатними приймати достатньо води для росту. Кожен вид рослин, природно, містить різні рівні кореневих солей. Якщо концентрація солоності в ґрунті досить висока, рослина в’яне і загине, незалежно від кількості поданої води. Датчик вимірювання вологи на рисунку 1.5.

Здатність вимірювати електромагнітну енергію на різних довжинах хвиль при взаємодії з матеріалом формує основу дистанційного зондування та спектральної науки. Фізичні характеристики матеріалу призводять до того, що електромагнітна енергія відображається, заломлюється або поглинається унікальним для кожного матеріалу способом.

Рисунок 1.5 – Датчик для вимірювання рівня солоності грунту

Ці взаємодії вимірюються на дискретних ділянках спектра, які, будуючи графік, утворюють унікальну форму, яка також відома як спектральна сигнатура матеріалу. Рослини взаємодіють з сонячною радіацією не так, як інші природні матеріали. Спектр рослин, як правило, поглинає червону та синю довжини хвиль, відображає зелену довжину хвилі, сильно відображає довжину хвилі ближнього інфрачервоного випромінювання (NIR) та демонструє сильні особливості поглинання на довжинах хвиль, де присутня атмосферна вода. Різні рослинні матеріали, вміст води, пігмент, вміст вуглецю, вміст азоту та інші властивості викликають подальші зміни в спектрі. Вимірювання цих варіацій та вивчення їх взаємозв’язку може надати значущу інформацію про стан рослин, вміст води, екологічний стрес та інші важливі характеристики. Ці взаємозв'язки часто описуються як індекси рослинності (VI)[5], порівняння двох показників на рисунку 1.6.

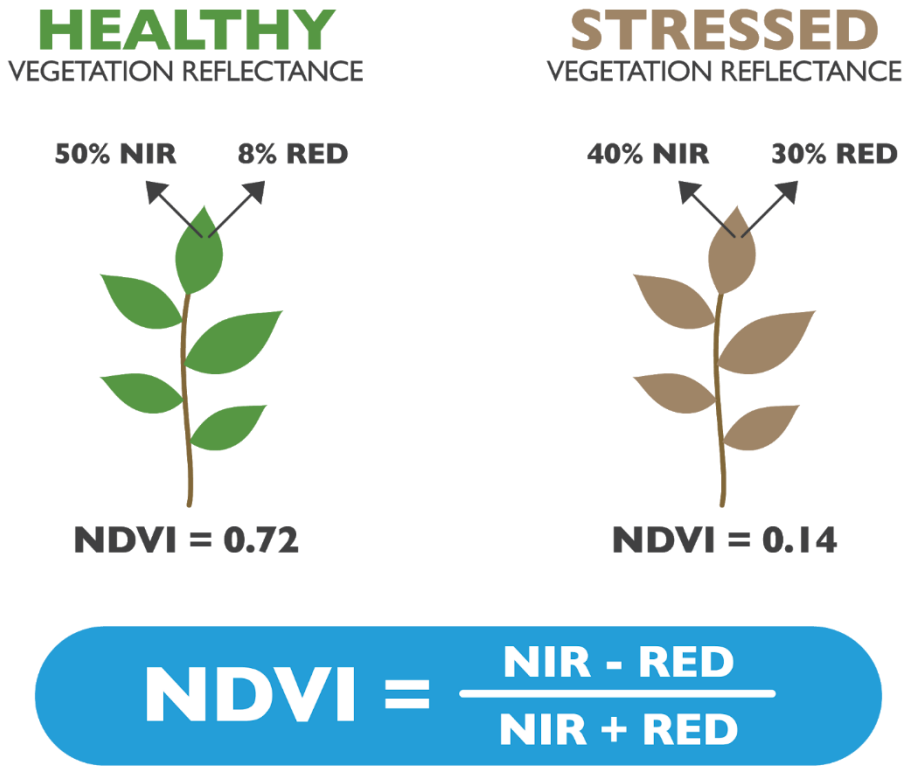

Рисунок 1.6 – Порівняння показників NDVI двох рослин

NDVI часто використовують для кількісного вимірювання рівня здоров'я, покриву та фенології рослинності (стадія життєвого циклу) на великих територіях.

Нормалізований індекс різницевої рослинності (NDVI) використовує співвідношення між ближнім інфрачервоним та червоним світлом у електромагнітному спектрі. Для розрахунку NDVI використовується формулу, де NIR – ближнє інфрачервоне світло, а червоний – червоне світло. Для ваших растрових даних ви берете значення відбиття в червоній та ближній інфрачервоних смугах.

На рисунку 1.7 зображено NDVI після холодної зими, що описує недостатньо оптимальні умови для підтримки життєдільності рослин, у контексті огляду навколишнього середовища як найважливішого чинника.

В останні роки у сільському господарстві почалося впровадження інтернет-технологій, супутникового зв'язку і геопозиціонування, робототехніки, датчиків і систем автоматизації. На сьогоднішній день в землеробстві вже впроваджені і активно використовуються системи GPS, що дозволили підвищити точність руху

техніки по полю і дали розвиток технології контрольованого проїзду по полю, а в перспективі – забезпечать перехід до роботизованого транспорту.

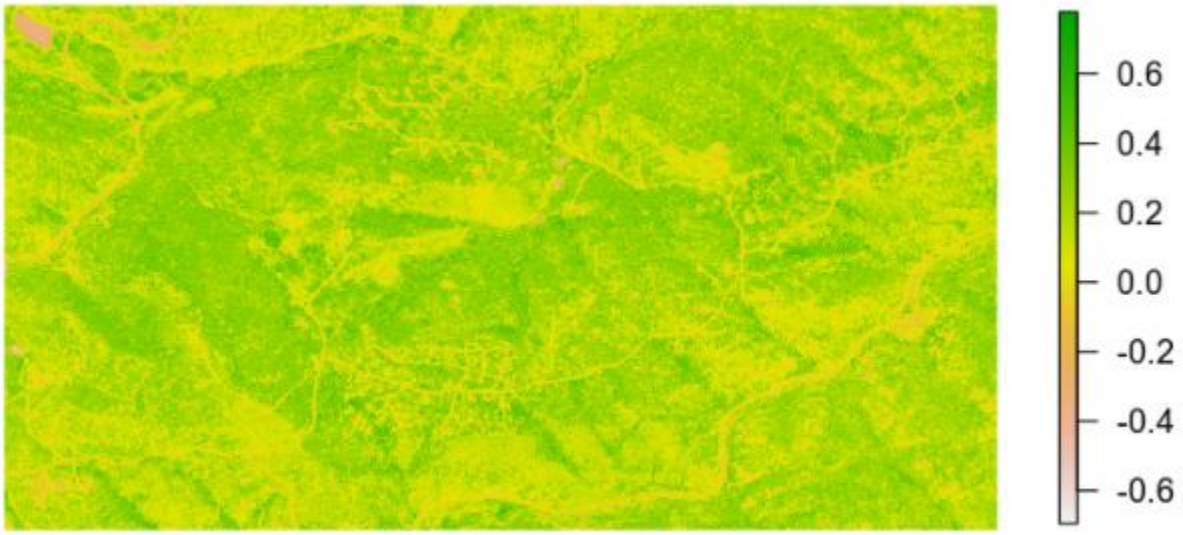

Рисунок 1.7 – Рівень NDVI при несприятливих погодних умовах

Використання чутливих датчиків дозволяє стежити за станом ґрунту, точно враховувати його склад на конкретних ділянках і вносити добрива там, де потрібно. Використання таких технологій дозволяє отримувати об'єктивну інформацію з полів про стан посівів, контроль комах, температурний режим грунту і інформацію про погоду. Використання датчиків і сенсорів в сільськогосподарській діяльності – важливий крок на шляху до створення інтелектуального сільського виробництва. Вони можуть безперервно передавати інформацію про стан контрольованих об'єктів - зокрема, значення таких параметрів, як вологість, температура ґрунту і повітря, рівень здоров'я рослини і т.д. Наприклад, основою системи визначення характеристик ґрунту є сенсори, які встановлюються в контрольних точках. Ці датчики призначені для виявлення неоднорідності рельєфу, типу ґрунтів, освітленості і т.д.. Вимірювані параметри відправляються на сервер, а звідти – на пристрої користувачів. Найпоширеніші датчики в сільському господарстві – це сенсори вологості ґрунту, їх застосовують сільгоспвиробники, які вирощують культури на зрошенні. Як правило, такі сенсори підключені до веб-сервісів для своєчасної передачі інформації. Зазвичай, при ручному поливі, норма витрати води розраховується завчасно і не враховує

багатьох параметрів, в результаті чого, через надмірну циркуляції води, може виникнути ерозія ґрунту. Датчики ж можуть виявити, при обліку таких чинників, як тип агрокультури, фаза її зростання і т.д., коли ґрунтовий шар досить зволожений і уникнути його перезволоження. У цьому випадку значно скорочується витрата води.

Використання чутливих датчиків дозволяє стежити за станом ґрунту, точно враховувати його склад на конкретних ділянках і вносити добрива там, де потрібно. Активно використовуються метеостанції для спостереження за посівами, розпилення добрив і засобів від комах. Використання таких технологій дозволяє отримувати об'єктивну інформацію з полів про стан посівів, контроль комах, температурний режим ґрунту і інформацію про погоду.

Враховуючи фактори того, що існує проблема зміни клімату через перевищення концентрації CO2 у атмосфері, використання такого датчика може позитивно вплинути як на якість продукції, так і на екологічну ситуацію.

Одним з важливих факторів розвитку рослини є рівень вуглекилого газу, який безпосередньо впливає на критичні життєвонеобхідні показники рослин, Сенсор який вимірює CO2 показано на рисунку 1.8.

Використання датчиків і сенсорів в сільськогосподарській діяльності – важливий крок на шляху до створення інтелектуального сільського виробництва. Вони можуть безперервно передавати інформацію про стан контрольованих об'єктів – зокрема, значення таких параметрів, як вологість, температура ґрунту і повітря, рівень здоров'я рослини і т.д.

Наприклад, основою системи визначення характеристик грунту є сенсори, які встановлюються в контрольних точках. Ці датчики призначені для виявлення неоднорідності рельєфу, типу ґрунтів, освітленості і т.д.. Вимірювані параметри відправляються на сервер, а звідти – на пристрої користувачів. Найпоширеніші датчики в сільському господарстві – це сенсори вологості ґрунту, їх застосовують сільгоспвиробники, які вирощують культури на зрошенні.

Рисунок 1.8 – Датчик CO2

Як правило, такі сенсори підключені до веб-сервісів для своєчасної передачі інформації. Зазвичай, при ручному поливі, норма витрати води розраховується завчасно і не враховує багатьох параметрів, в результаті чого, через надмірну циркуляції води, може виникнути ерозія ґрунту. Датчики ж можуть виявити, при обліку таких чинників, як тип агрокультури, фаза її зростання і т.д., коли ґрунтовий шар досить зволожений і уникнути його перезволоження. У цьому випадку значно скорочується витрата води[6].

Розглянемо ризики використання автоматизації:

– приховані помилки та проблеми в прикладному програмному забезпеченні системи автоматизації не виявляються, доки вони не спричиняють проблем у процесі чи експлуатації;

– невідповідні дії оператора або, навпаки, оператори, які не вживають заходів, коли повинні;

– операційні процедури, які є помилковими або неповними, тому вони не використовуються.

З одного боку, традиційно пропоновані в галузі рішення важкі у використанні і вимагають великих налаштувань. Традиційні підходи зробили технічне обслуговування системи кошмаром. Інвестиції в оновлення традиційних підходів можуть бути дуже значними. Традиційні рішення рідко будуються для підтримки системи автоматизації та вдосконалення роботи.

Автоматизація допомагає досягти кривої зростання, підтримуючи більш високі врожаї. Автоматизована система поливу дозволяє вирощувати рослини
ближче одна до одної, займиючи менший простір, оскільки не потрібен простір для того, щоб людина дійшла до кожної окремої рослини. Автоматичні датчики мають не тільки ті дані, які можна зчитувати через заздалегідь визначені проміжки часу, але також показання цих датчиків зберігаються таким чином, що можна отримати більш високі результати для аналізу та діагностики, що призводить до вищих урожаїв та більш дружньої практики з навколишнім середовищем. У наш час на ринку представлені мікроконтролери, сумісні з широким спектром датчиків і можуть використовуватися для автоматичного моніторингу та робототехніки.

Автоматизація освітлення і поливу для внутрішніх і зовнішніх рослин дозволяє скоротити витрати на енергоспоживання до 40%, а також збільшити термін служби обладнання за рахунок їх оптимального використання.

Задля оцінки ефективності зрощування та доцільності використання відповідних технік та підходів агротехнік бере до уваги наступні дані про вегетативні та репродуктивні ознаки рослин:

– висота рослини, виміряна від крони рослини до максимальної точки зростання основної гілки;

– поперечний розкид, виміряний як максимальний діаметр рослини;

– максимальна довжина листка, виміряна від основи черешка до верхівки листа;

– максимальна ширина листа;

– наявність або відсутність квіткових бруньок;

– кількість колосків з чоловічими квітковими головками;

– довжина кисті чоловічої статі;

– виділення пилку, що контролюється як наявність або відсутність протягом часу;

– суха маса повітряної біомаси, виміряна в кінці періоду росту.

Розроблювана програмна система надаватиме кінцевому користувачеві змогу вносити нові корективи у вже існуючі сценарії або створювати нові для досягнення найкращих результатів зрощування. Сценарії представляють собою набір кількісних значень параметрів, які задають фактори росту, такі як вологість ґрунту, кислотність та інші, часові проміжки, у які ці параметри повинні бути замінені іншими значеннями.

Система налаштовується під конкретну рослину, тобто графіки поливу, додавання добрив не прив’язані до фізичного часу, а відштовхуються від життєвого циклу рослини і показників сенсорів. Цей підхід надає змогу зручно та ефективно налаштовувати систему та корегувати її роботу. Метою та завданням є багатокористувацький доступ і управління апаратним комплексом моніторингу та підтримки життєдіяльності рослин.

1.3 Аналіз відомих технічних рішень

Завдання автоматизації зрощування не є новим. Воно може бути вирішено шляхом моделювання мікроклімату, підбором необхідних параметрів. Існуючі програмно-апаратні рішення вимагають значних грошових вкладень і є складними у впровадженні. Це зумовлює певні економічні ризики. У цей час постає питання про доцільність та виправдання таких вкладень.

Треба зазначити, що наразі не існує прямих аналогів розроблюваної системи. Більшість з існуючих програмних рішень є вузькоспеціалізованими та сконструйованими під конкретні апаратні комплекси або приховані від загального доступу.

Рішення, які представлені на ринку, мають велику кількість недоліків та упущень. Такі системи не здатні до прогнозування майбутнього врожаю. Вони не

здатні до тонкого налаштування параметрів та зазвичай надають можливість для маніпулювання усіма даними одночасно. Це зумовлює відсутність можливості для поодиночного налаштування кожного з параметрів. Такі
програмні системи важко тестувати та при виявленні помилок майже неможливо визначити у якій частині системи проблема, апартній чи програмній зокрема. Наявні аналоги дають змогу підтримувати температуру та впливати на мінімум параметрів, що значно обмежує користувацький досвід.

Тож можна підсумувати наступні обмеження:

– відсутність можливості прогнозування;

– відсутність тонкого налаштування окремих параметрів;

– складність тестування;

– відсутність можливості диференціації помилок;

– необхідність наявності певних технічних знань;

– обмежений користувацький досвід.

Головною особливістю розробленого проєкта є наявність сценаріїв для визначення факторів, які впливають на ріст. Вони надають можливість кінцевому користувачеві задавати, спостерігати та контролювати параметри системи протягом усього життєвого циклу рослини. Дана модель є динамічною, легко адаптується до змін та зручна для впровадження. Далі буде розглянуто деякі з відомих рішень на ринку.

Пристрої Intelligrow доступні через веб-браузер, тому є можливість віддалено контролювати свій зростаючий врожай з телефону, планшета чи комп’ютера. Intelligrow забезпечує повну видимість 24 години на добу та дозволяє керувати кількома об’єктами, не виходячи з офісу, за допомогою єдиного входу. Рівень поживних речовин та pH керується автоматично, є можливість встановлення віддалених сигналів тривоги та реєстрування даних про свій прогрес. Для усіх налаштувань та даних щодня створюються резервні копії на захищених серверах, що дозволяє швидко відновити їх у разі несправності, зменшити ризик втрати даних та зберегти всі налаштування конфігурації.

На рисунку 1.9 – додаток Intelligrow від компанії Autogrow.

Рисунок 1.9 – Користувацький інтерфейс Intelligrow

Недоліком такої системи є необхідність встановлення відповідного апаратного забезпечення, його придбання та вбудовування у існуючий об’єкт. Крім коштів для придбання даної системи, користування потребує щомісячної плати у розмірі 32 доларів. Також така система не підходить для невеликого об’єму продукції та розрахована на використання великими підприємствами спрямованими на оптову продукцію.

Програмне забезпечення GroLab є кабіною управління програмно-апаратною системою. Основна мета цього програмного забезпечення – дозволити повністю налаштувати всю систему, надаючи всі інструменти для налаштування модулів та налаштувань пристроїв, що відповідають функціоналам будь-якої системи вирощування сільського господарства. Інтерфейс додатку GroLab зображено на рисунку 1.10.

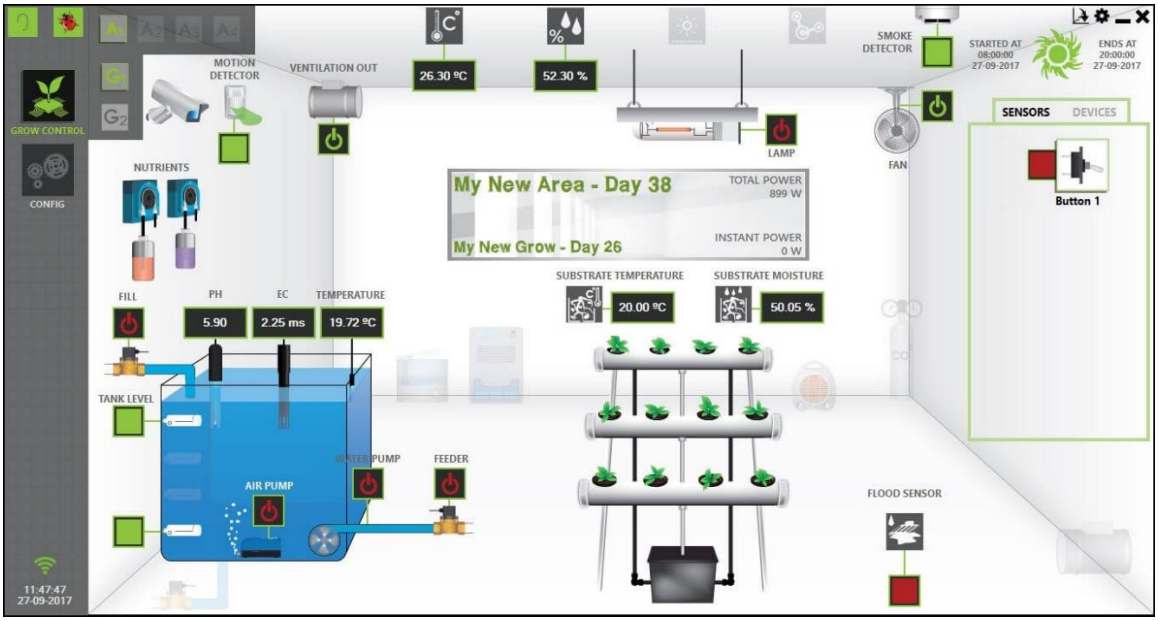

Рисунок 1.10 – Користувацький інтерфейс GroLab

Є можливість відстежування розвитку за допомогою IP-камер та варіантів введення даних, включаючи графіки, історичні дані та тенденції. Всі дані датчиків, виконавчих механізмів, можна легко експортувати у зручний файл для зовнішнього глибокого аналізу. Використовуючи сторонній пристрій, здатний реєструвати хмарні дані, система GroLab може періодично завантажувати дані на цей пристрій у форматі JSON. GroLab має кілька шарів для захисту даних, крім облікових даних, необхідних для доступу до програмного забезпечення GroLab, система також вимагає облікові дані для доступу до GroNode (основного модуля системи). Програма дозволяє створювати графіки для зрошення, освітлення. На сайті представлений пробний доступ до програмного забезпечення, який не потребує придбання модулів GroLab. Це забезпечує наявність демонстраційного режиму доступу до програми.

Недоліками такої системи є необхідність придбання апаратних модулів разом із програмним забезпеченням для роботи. Також програма доступна лише у декількох мовах, та не має потрібної локалізації.

Характеристика продуктів-аналогів представлена у таблиці 1.1.

Таблиця 1.1 – Характеристики продуктів-аналогів

	Intelligrow	GroLab
Платне ПЗ та додаткові модулі	+	+
Веб-версія додатку	+/–	–
Простий користувацький інтерфейс для користувача без спеціальних технічних знань	-	–
Можливість керування пристроєм за допомогою віддаленого доступ	_	+
Побудова багатоетапних сценаріїв розвитку	–	–

1.4 Висновки

Проаналізувавши існуючі рішення було сформовано актуальність роботи, визначено ключові завдання та конкретизовано задачі. Актуальність роботи полягає в створенні конкурентноздатного продукту, ключовою особливістю якого є можливість детального налаштування сценаріїв розвитку рослин враховуючі фактори, які безпосередньо впливають на процеси життєдіяльності рослин, яка є важливою складовою для подібного класу систем та відсутня у системах конкурентів.

Основними напрямками цієї роботи, виходячи з наведеного аналізу предметної області та проведеного порівняння існуючих рішень з врахуванням виявлених

недоліків, є реалізація власного рішення на основі агрегації інноваційних рішень існуючих систем та зведення до мінімуму недоліків та реалізація набору власних рішень. Серед яких побудова сценаріїв життєвого циклу рослин на основі набору наведених критеріїв. Функціональна можливість обміну існуючими з сценаріями для уніфікації процесу розвитку рослин в межах груп користувачів. А також надання можливості автоматичної корекції параметрів, що впливають на життєвий цикл рослини на основі існуючих сценаріїв і їх опублікованих якісних характеристик з метою спрощення управління процесом розвитку поточної рослини, що забезпечує низький поріг входу по критерію якісної оцінки поточних знань людини з даної предметної області, як наслідок кінцевим споживачем зможе стати людина без специфічних технічних знань.

2 ПРОЕКТУВАННЯ СИСТЕМИ

Після того як проведено аналіз та обґрунтовано необхідність виконання даної роботи, необхідно провести проектування реалізації програмного продукту, основою для якого є декомпозиція переваг конкурентів в вигляді функціональних вимог, огляд доступних технічних рішень, що відповідають критерію оцінки комплексної системи за критерієм вартості, який в свою чергу не повинен перевищувати вартість конкурентів.

2.1 Концептуальна модель програмного продукту

Для опису роботи системи необхідно побудувати концептуальну модель системи, що розробляється. Така модель повинна містити в собі опис всіх учасників процесів системи. Базуючись на огляді предметної області, а саме аналогів даного класу систем в межах огляду веб орієнтованих рішень було вирішено використовувати дві ключових ролі користувача та адміністратора, оскільки зменшення кількості ролей для даної системи унеможливлює реалізацію всіх формальних вимог, відповідно збільшення кількості ролей призводить до ускладнення процесу управління апаратним комплексом, що в свою чергу порушує критерії до власної розробки.

Розглянемо ролі акторів:

- адміністратор – фахівець, який буду керувати усіма процесами у системі, і він же буде мати повний доступ до всіх функцій;
- користувач – людина, яка може під’єднати апаратний комплекс до системи, налаштовувати його та використовувати його для зрощення рослин.

Відповідно до наведених в аналізі предметної області критеріїв, та в наслідок проведеного аналізу існуючих рішень конкретизуємо формальні вимоги яким повинна відповідати розроблювана система в межах наведених ролей.

Розглянемо прецеденти користувача:

– реєстрація у системі;

– з’єднання апаратного комплексу з системою керування за допомогою унікального ідентифікатора;

– первісне калібрування системи та налаштування основних компонентів контролю над розвитком рослини;

– вибір попередньо встановленого сценарію для потрібної рослини;

– власноручне корегування налаштувань у сценарії;

– створення та збереження власних сценаріїв;

– відстежування стану рослини у режимі реального часу;

– перегляд статистики розвитку рослини за обраний період;

– можливість перегляду оприлюднених сценаріїв інших користувачів системи.

Розглянемо прецеденти адміністратора:

– перегляд та редагування налаштувань користувачів;

– можливість втручатися у роботу системи користувача та налаштовувати апаратний комплекс певного користувача;

– створення та редагування сценаріїв життєвого циклу рослин;

– перегляд активних сесій апаратних комлексів;

– реєстрація користувачів;

– перегляд повідомлень про системні помилки.

Відповідно до стандартизації процесу оформлення документації до програмних засобів серед яких є веб орієнтовані системи для відображення функціональних вимог було використано діаграму прецедентів стандарту UML 2.0[9]. Результати побудови функціональних вимог в відповідності до стандартизації наведені в додатку А.

2.2 Архітектурне рішення

Для реалізації поставлених функціональних вимог, наявних до даного класу систем, необхідно визначити загальну архітектуру даної системи.

З проведеного огляду існуючих рішень, за різновидністю рівнів комунікації реалізацію даних вимог можна поділити на управління через віддалений доступ та локальне.

Базуючись на функціональному критерію віддаленого управління, реалізація підходу з локальним управлінням не є актуальною та суперечить даним вимогам.

За критерієм комунікації підхід з наявністю серверу управління та обробки клієнтських запитів відповідає поставленим вимогам. Проведено огляд існуючих архітектур: SOA, клієнт-серверна. Відповідно SOA архітектура вимагає наявність засобів масштабування, передбачає значні витрати на підтримку інфраструктури та породжує проблеми консистентності даних, що є критичним для даного класу систем[7]. Клієнт – серверна архітектура є оптимальною за співвідношенням критеріїв складності реалізації, простоти використання, та ресурсовитратності, передбачає з'єднання з пристроєм засобами протоколів комунікації що відповідають стандартизації та базуються на моделі OSI[8].

В межах реалізації клієнт – серверної архітектури доступ до системи може бути здійсненим використовуючи веб-браузер, під'єднаний до мережі Інтернет.

Основними елементами, що визначені в межах обраної по вказаним критеріям архітектури, є серверна, клієнтська та рівень комунікації, що забезпечує передачу інформації між компонентами даної архітектури. Важливим аспектом поточного класу систем є забезпечення взаємодії і відображення даних в часі, наближеному до реального, що забезпечує можливість своєчасного втручання користувача в процес розвитку і, відповідно зміну сценарію, розвитку поточної рослини. Клієнтська частина надає користувацький інтерфейс системи, що забезпечує візуалізацію процесу управління апаратним комплексом та реалізує всі необхідні описані функціональні вимоги з точки зору взаємодії з користувачем, створює інтерфейс взаємодії в реальному часі з серверним компонентом, обробляє дані вводу користувача та зберігає їх за необхідності в пам'яті поточного пристрою для оптимізації взаємодії по часовому критерію. Серверна частина відповідає за реалізацію збереження даних та взаємодію користувача та апаратного комплексу. В свою чергу для реалізації такої взаємодії необхідно обробляти за допомогою каналів комунікації дані клієнтської частини з використанням реалізованих бізнес процесів в межах рівня бізнес логіки серверної частини, реалізуючи збереження даних за допомогою рівня доступу до даних на рівні серверного компоненту та джерела даних

як окремої одиниці архітектури даної системи. Для відповідей на запити у необхідному форматі в межах проектування поточної системи було проведено аналіз форматів даних для комунікації та обрано JSON за критерієм обсягу мета- інформації, що передається.

В відповідності до наведеного опису, реалізація даної системи буде відбуватись на основі клієнт-серверної архітектури. Відповідно до стандартизації оформлення документації до програмних продуктів, обумовленої в стандарті UML 2.0[9]. Результат проектування архітектури наведено на діаграммі розгортання системи у додатку Б.

Аргументація вибору архітектури.

Система, що розглядається, буде працювати у режимі наближеного до реального часу, рівень комунікації буде представлено в вигляді двох каналів зв'язку WebSocket, що забезпечує обмін інформацією в наближеному до реального часі з апаратним комплексом у форматі даних JSON, загальна архітектура буде представлена в вигляді багатокомпонентної клієнт-серверної архітектури.

Проаналізувавши усі функціональні вимоги до системи та оглянувши основні архітектури веб-додатків, було обрано багатокомпонентну архітектуру, що буде складатися з джерела даних, серверного компоненту, апаратного комплексу та клієнтської частини. Комунікація між складовими компонентами архітектури повинна забезпечити можливість оповіщення в наближеному до реального часу повідомленнями в межах взаємодії серверу з апаратним комплексом та клієнтським компонентом.

2.3 Рівень комунікації

Відповідно до специфікації вимог до системи існує завдання на проектування рівня комунікації, що повинен відповідати вимогам здійснення обміну повідомленнями між сервером, апаратним комплексом і клієнтським представленням в режимах наближеного до реального часу та за вимогою компоненту. Некоректне вирішення даного завдання призводить до часових затримок в межах комунікації та

накопичування великого обсягу необроблених запитів на кожній одиниці компонентної одиниці в архітектурі. Прикладом невдалої реалізації цього механізму є розглянуті в аналізі існуючих рішень системи, що реалізовані на основі клієнт-серверної архітектури. Для вирішення даної задачі існують наступні способи: оновлювати дані виключно після безпосереднього запиту користувача, використовувати Long-Polling[10] (веб-клієнт виконує запити на сервер з певним інтервалом) або використовувати відмінний від HTTP протокол для обміну повідомленнями між браузером і веб-сервером в режимі часу, наближеного до реального (наприклад, WebSocket).

В даному веб-додатку ця проблема стає достатньо гостро, бо користувач має бачити стан пристрою та отримувати якомога актуальнішу інформацію.

В першу чергу, як було сказано вище, організацію поновлення візуальних даних у веб-клієнті можна виконати за допомогою звичайних HTTP-запитів за вимогою користувача, що не забезпечить належний час доставки в межах функціональних вимог до поточного класу систем.

По-друге, організацію поновлення візуальних даних у веб-клієнті можна виконати за допомогою Long-Polling. У цьому випадку використовується той же HTTP-запит даних, як і в першому випадку, проте він повторюється з деяким інтервалом часу, таким чином, веб-клієнт має можливість показувати інформацію в часі, наближеному до реального.

Даний метод простий у реалізації і може використовуватися при розробці простого веб-інтерфейсу мережевого пристрою для моніторингу невеликої кількості даних, наприклад, моніторинг однієї таблиці параметрів мережевого пристрою в реальному часі.

Нарешті, оновлення візуальних даних можна реалізувати за допомогою WebSocket. В цьому випадку веб-клієнт відкриває з'єднання з сервером і підписується на необхідні теми, після чого сервер відправляє дані по підписці в той момент, коли вважатиме за потрібне, наприклад, коли дані змінилися в базі даних. Прикладом веб-додатка може стати будь-який веб-додаток з оновленням даних в реальному часі, так

як WebSocket є найефективнішим з точки зору продуктивності і навантаження на сервер способом комунікації у часі, наближеного до реального.

Таблиця 2.1 – Порівняння способів організації оновлення даних у веб-клієнті

Критерій порівняння	Оновлення тільки після запиту	Поллінг	WebSocket
Можливість створення додатка з оновленням даних в реальному часі	Ні	Ні	Так
Необхідність реалізації методів оптимізації великого потоку даних на стороні веб-клієнта	- (великий потік даних може виникнути тільки через велику кількість прямих запитів користувачів, його потрібно обробляти на сервері, внаслідок чого уповільнення Роботи інтерфейсу у інших користувачів не виникне)	+ (Long-Polling використовується для реалізації програми з оновленням даних в реальному часі, тому дані потрібно оновлювати досить часто. Однак, якщо на сервері даних дуже багато (вся база може важити близько 1 ТБ), то	+ (Сервер сам посилає дані на веб-кліент, тому, якщо сервер спостерігає в базі даних часті зміни - все зміни потраплять до веб-кліенту у вигляді великого потоку повідомлень.)

		Long-Polling, навіть з реалізацією кластеризації даних може стати абсолютно непридатним рішенням.)	

З порівняння в таблиці 2.1 видно, що для обробки великих даних найвигідніше використовувати WebSocket, тому що розмір потоку повідомлень між веб-клієнтом і сервером виходить найменшим і найбільш обґрунтованим (тому що метод оновлення тільки після запиту користувача не підходить для створення додатка з даними, що оновлюються в реальному часі, а Long-Polling не підходить для постійного завантаження даних, коли даних стає дуже багато, до того ж, при використанні Long-Polling веб-клієнт виконує запити на сервер завжди, навіть тоді, коли дані в базі даних не змінилися. Явним недоліком при використанні WebSocket стає безліч різних так званих «обгорток» над протоколом. В такому випадку необхідно вибрати такий протокол, який буде підтримуватися і на стороні веб-клієнта, і на стороні сервера.

Базуючись на наведеному огляді та аналізі вимог до даного класу систем в межах комунікації та обміну повідомленнями між сервером і апаратним комплексом та сервером і користувачем в межах відображення критично важливої інформації на яку необхідно своєчасно реагувати, доцільно обрати WebSocket для реалізації видачі не критичних по часу повідомлень. Рівень комунікації в межах сервер – клієнт потрібно будувати на основі REST API, бо немає критичних до часу вимог, детальна схема у додатку В.

2.4 Джерело даних

В межах описаної архітектури необхідно обрати тип джерела даних. Дане завдання базується на аналізі логічних моделей даних та аналізі існуючих технологій, що забезпечують ефективну реалізацію операцій CRUD за критерієм швидкодії для тих чи інших типів зв’язків та специфікацій моделей.

Виходячи з специфікації типу системи варто відзначити, що дані того чи іншого датчику можуть складатись як з одного значення, так і з набору значень, датчики на апаратному комплексі можу бути об’єднані в групи та відправляти дані моніторингу з одним ідентифікатором та багатьма значеннями.

Базуючись на цих критеріях логічної моделі постає вибір між використанням атрибутивної моделі в межах реляційної бази даних, та використання документів в межах NoSQL бази даних.

З урахуванням специфіки роботи реляційних баз даних можна відмітити не оптимальність побудови рекурсивних запитів та складність реалізації зберігання динамічної кількості полів, що досягається реалізацією атрибутивної моделі, яка в свою чергу збільшує кількість таблиць та ускладнює зв’язки між ними. До того ж, базуючись на функціональній вимозі про об’єднання датчиків в межах відправки інформації і, як наслідок її зберігання в дочірніх елементах, маємо рекурсію в межах атрибутивної моделі.

Базуючись на частоті звернень для отримання інформації по датчикам – як одному із ключових елементів CRUD операцій в межах даної системи, отримуємо складний план виконання запиту, зниження продуктивності та складність реалізації обробки даних.

Розглянемо NoSQL бази даних. Ця архітектура спрямована на зберігання інформації в вигляді JSON документа, що в свою чергу забезпечує підтримку масивів даних, спрощення зберігання рекурсивних об’єктів та, через відсутність типізації документу, можливість зберігання різного набору даних[11].

В відповідності до наведеного порівняння було прийнято рішення про використання NoSQL бази даних.

2.5 Серверний компонент

Вибір технології реалізації серверного компоненту.

Існує безліч варіацій технологій, що дозволяють реалізувати серверний компонент застосунку в межах клієнт-серверної архітектури. Сформулюємо вимоги до технології. Технологія повинна забезпечувати гнучкий процес обробки даних, побудоване на ній рішення повинно бути масштабованим. Наявність нетипізованих даних в межах логічної моделі вимагає відсутності чіткої типізації в межах мови всередині технології. Наявність складних бізнес процесів вимагає наявності можливості задання бізнес-правил через функціональний підхід. З урахуванням вимог до рівня комунікації, технологія повинна забезпечувати ефективну за критерієм швидкодії обробку і реакцію на події та підтримувати взаємодію через вказані інтерфейси комунікації. Також технологія не повинна вимагати великих грошових затрат на ліцензії з урахуванням критерію загальної вартості продукту та порівняння з аналогами. Також час розробки, повинен бути мінімальним, оскільки людино-години також збільшують вартість продукту. А з урахуванням специфіки поточного класу систем технологія повинна забезпечувати високий рівень швидкодії серверного компоненту.

Технологія Node.js є оптимальною за критеріями швидкодії та відсутності чіткої типізації[12]. Розглянемо дану технологію детальніше та визначимо її переваги в межах застосування для даного класу систем. Node.js, завдяки подієво-орієнтованій архітектуру, підходить для реалізації додатків реального часу, бо технологія повідомлень реалізується за допомогою WebSocket. Ініціація зв’язку може відбуватися як з боку серверу, так і зі сторони клієнтської частини. Ця технологія відрізняється від стандартної парадигми, коли зв’язок ініціюється тільки клієнтською частиною.

Порівнюючи Node.js зі стандартними веб-сервісами, де новий потік створюється кожним новим зверненням, що має великий вплив на оперативну пам’ять, Node.js обробляє дані в одному потоці, використовуючи неблокуючий ввод-вивід та підтримує багату кількість з’єднань, які розташовуються у подієвому циклі.

Аргументація вибору технології взаємодії з джерелом даних.

Базуючись на аргументованій технології створення серверного застосунку, виникає завдання вибору фреймворку по взаємодії з існуючою базою даних. Існує два ключових підходи реалізації цього завдання. Побудова рівня взаємодії за допомогою встановлення з'єднання з джерелом даних на пряму та передача команд на виконання. Перший підхід є ресурсозатратним, а також в процесі розробки створює складнощі по підтримці консистентного стану джерела даних на різних серверах, наприклад, оточені розробника та користувацькому інтерфейсі.

Інший підхід полягає у використанні додаткового шару абстракцій, що містить перший рівень кешування та забезпечує підтримку версійності бази даних і спрощує роботу з нею.

Виходячи з вимог до програмного продукту було обрано варіант з використанням ODM. В межах даного підходу самим популярним фреймворком по роботі з MongoDB є Mongoose. Виходячи з інформації наведеною у джерелі[13] за критерієм продуктивності та скороченню часу обробки.

Mongoose – це ODM, яка, у відповідності стандартам MongoDB, описує об'єкти зі строгою типізацією.

Використовуючи Mongoose, розробнику надається багато функціональних можливостей щодо роботи з даними. Mongoose реалізує приведення значень і типів джерела даних до DAO об'єктів.

MongoDB надає можливість розробнику власноручно встановити особливості встановлення та перетворення даних, що позитивно впливає на швидкість та простоту обробки розширених та нетипізованих моделей, які притаманні даному класу систем[14].

Посилання на інші колекції у базі даних зберігається у типі даних ObjectId. Наприклад, якщо є колекція сценаріїв розвитку рослин і колекція користувачів, документ сценарію містить властивість ObjectId, що має посилання на певного автора сценарію.

Масиви зберігаються у типі даних Array з можливістю виконання операцій роботи над масивами, як в JavaScript, наприклад, map, reduce, push, every, some.

2.6 Клієнтська частина

Базуючись на функціональних вимогах до поточної системи, інтерфейс користувача є складним, в межах побудови UX зв’язків між дочірніми компонентами представлень. Варто відмітити набори представлень, що дублюються на різних сторінках. Вказані вище елементи вимагають вирішення завдання побудови гнучкого користувацього компоненту системи за критерієм розширюваності та перевикористання шаблонних елементів з завершеною логічною поведінкою. Також поточна система вимагає реалізації завантаження складних моделей міжкомпонентної взаємодії елементів, що в свою чергу унеможливлює відсутність використання MV* шаблонів для розбиття клієнтської логіки та представлення. Через наявність різних станів кожного окремо взятого часткового клієнтського представлення виникає необхідність застосування підходу що включає в себе можливість реалізації даних переходів. Одним із таких підходів є SPA. Розглянемо даний підхід детальніше.

На сьогодні, односторінкові додатки (SPA) стали стандартом при розробці клієнтських частин веб-застосувань. Щоб написати повноцінний односторінковий додаток, не використовуючи фреймворк, а лише чистий JavaScipt, потрібно витратити значно більшу кількість часу, реалізація архітектури додатку за принципами SOLID буде значно складнішим завданням[15]. Аналізуючи ці фактори вирішено використовувати фреймворки SPA.

Огляд технологій розробки SPA.

Розглянемо три найактуальніших фреймворки, виділивши їхні переваги й недоліки.

Angular – був розроблений компанією Google, ставши послідовником фреймворку AngularJS. Він написаний на мові TypeScript, який в свою чергу є надбудовою над JavaScript.

Розглянемо переваги Angular:

– структура Model-View-Whatever, за якою застосування на Angular складається з трьох пов’язаних компонентів, надаючи можливість декомпозиції елементів системи[16];

– висока читабельність шаблонів;

– прив’язка подій та властивостей;

– це повноцінний фреймворк, який надає усі необхідні інструменти для реалізації складного веб-додатку;

– велика спільнота.

Розглянемо недоліки Angular:

– відсутність Shadow DOM для інкапсуляції HTML атрибутів та стилів;

– відсутність Virtual DOM для можливості повторного рендерингу окремих елементів;

– стратегія виявлення змін при стандартному режимі обновлює усе DOM-дерево, що призводить до втрат продуктивності;

– великий об’єм коду через розподілення відповідальності на окремі модулі.

React – бібліотека JavaScript, призначена для розроблення користувацького інтерфейсу для веб-застосувань. React - продукт компанії Facebook, саме тому React набув великої популярності у спільноті розробників клієнтських веб-застосувань[17].

Розглянемо переваги React:

– написання логіки компонентів пишеться на JavaScript без використання шаблонів;

– присутність Shadow DOM;

– представлення реагує на будь-які зміни у компоненті за допомогою односторонньої прив’язки даних;

– pure functions, які використовуються для збереження стану додатку у форматі Immutable;

– React Native надає можливості створення гібридних мобільних додатків, використовуючи код React, з великою продуктивністю;

– велика спільнота.

Розглянемо недоліки React:

– це не повноцінний фреймворк, тому сторонні модулі та бібліотеки необхідні для розробки веб-застосувань;

– немає чіткої структури додатку, як наслідок, побудова чистої архітектури стає більш складним завданням.

Vue.js має велику динаміку зростання у спільноті. Його було створено колишнім співробітником Google, який розробляв AngularJS. Він має низький поріг входу та відносну просто у створенні веб-інтерфейсів[18].

Розглянемо переваги Vue.js:

– структура Model-View-Whatever, для побудови чистої архітектури;

– мала кількість коду;

– шаблони пишуться без використання JSX;

– Virtual DOM для можливості повторного рендерингу окремих елементів[19];

– велика популярність у спільноті, через відсутність необхідності вивчати нові технології та мови програмування, на відміну від Angular;

– у порівнянні з React та Angular – проста в вивченні.

Розглянемо недоліки Vue.js:

– не повноцінний фреймворк, а при малій кількості бібліотек до Vue, можливо зіштовхнутися з труднощами при розробці складних додатків.

Аргументація вибору технології клієнтської частини.

Розглянувши та проаналізувавши три основних фреймворки SPA, було зроблено висновки, що Angular надає можливість створювати більш чистий код, з точки зору на архітектуру, притримуючись принципів SOLID[20].

Також треба зазначити, що Angular є повноцінним фреймворком ,у якому надається велика кількість різноманітних можливостей, щоб повноцінно розпочати розробку веб-додатка одразу після встановлення. Не має потреби встановлювати сторонні бібліотеки, щоб реалізувати базовий функціонал веб-додатку системи керування апаратним комплексом.

Архітектура, побудована на компонентах, надає широкий набір можливостей для подальшого масштабування, повторне використання існуючих функціональних блоків, тестування та створення додаткового, нового функціоналу більш простим.

Враховуючи усі переваги Angular, саме цей фреймворк було обрано для реалізації клієнтської частини.

2.7 Висновки

В даному розділі було наведено огляд існуючих архітектур, що властивий даному класу систем, наведено переваги застосування в межах декомпозиції недоліків застосованих архітектур в рішеннях конкурентів та визначено оптимальну архітектуру для поточного додатку за критерієм масштабованості. Описано технологічні рішення, для вибору безпосередніх технологій для реалізації компонентів системи, таких як сховище даних, серверний та клієнтський компоненти. В якості сховища даних було обрано NoSQL базу даних через формат даних, що будуть збережені та їх розширюваність в межах логічної моделі. В порівнянні з реляційною БД, що здатна реалізовувати поточні задачі та забезпечити шаблонність рішення на такому самому рівні на основі атрибутивної моделі, що програє за критерієм швидкодії[14]. В якості технології серверного компоненту системи було обрано NodeJS за критеріями високої продуктивності та подієво-орієнтованої архітектури, використовуючи яку буде побудовано додаток реального часу.

В якості технології клієнтського компоненту системи було обрано Angular за критеріями об’єктно-орієнтованої архітектури, яка сприяє масштабованості та дотриманню SOLID принципів при розробленні застосування.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

На основі наведеного огляду та проведеного аналізу існуючих архітектур та технологій було визначено основні компоненти системи та технології для реалізації даних компонентів. В межах поточного розділу буде проведено огляд декомпозиції складових компонентів архітектури та їх реалізації.

3.1 Реалізація архітектурного рішення

Рівень сервера додатків.

Рівень сервера додатків буде реалізований за допомогою технології NodeJS та фреймворку Express.

Даний рівень є сполучною ланкою між рівнями клієнта і базою даних, в якому зберігаються велика частина бізнес-логіки. На цьому рівні в розроблюваній системі міститиметься:

– бізнес-логіка для прийому і обробки запитів від клієнта;

– створення колекцій для бази даних;

– бізнес-логіка роботи з апаратним комплексом;

– бізнес-логіка взаємодії з базою даних.

Рівень апаратного комплексу.

Даний рівень обробляє дані, які надає сервер додатків та у режимі реального часу відправляє актуальну інформацію про стан апаратного комплексу для подальшого відображення та можливості редагування необхідних параметрів зі сторони клієнтської частини.

Рівень бази даних.

Рівень бази даних, реалізований за допомогою технології MongoDB та інструменту роботи з даними Mongoose.

Даний рівень буде здійснювати зберігання та обробку даних, а також взаємодіяти виключно з рівнем сервера додатків.

Рівень клієнтською частини.

Рівень клієнтською частини буде побудовано використовуючи фреймворк Angular, клієнтська частина за допомогою REST і WebSocket буде отримувати та передавати необхідні дані, щоб реалізувати клієнтську бізнес-логіку та відображати усю необхідну для успішної роботи застосування інформацію.

Для уявлення взаємозв'язку між логічними і фізичними модулями, необхідно створити схему компонентів.

Компонент «Сервер додатку» – реалізує бізнес-логіку для прийому і обробки запитів від клієнта, надає інтерфейс IServer і отримує інтерфейс IDatabase.

Компонент «Апаратний комплекс» – отримує необхідні дані з серверу та опрацьовую їх на апаратному рівні.

Компонент «СУБД» – реалізує доступ до даних і надає інтерфейс IDatabase.

Компонент «Клієнт» – отримує інтерфейс IServer, складається з двох роздільних частин:

Структурну схему компонентів системи більш детально розглянуто у додатку Г.

3.2 Реалізація рівня комунікації

Клієнт серверна комунікація.

Базуючись на функціональних вимогах системи, обрано два типи комунікації. Для обміну терміновими повідомленнями – WebSocket.

Побудова комунікації між клієнтом та сервером реалізується за допомогою WebSocket.

Зі сторони клієнтської частини буде відкрито з’єднання, при його ініціалізації на рівні клієнтської частини методі ConnectorService connect() передається унікальний токен користувача, ім’я події, яка в даному випадку має назву init, з серверної сторони оброблюється запит з’єднання у методі SocketService onConnect(), перевіряючи токен користувача методом AuthService checkAuth() та повертаючи клієнтській частині повідомлення про успішний статус з’єднання, схему взаємодії у реальному часі продемонстровано у додатку Д.

Для створення WebSocket-з'єднання на стороні серверу потрібно скористатися відповідним конструктором:

```
const mainUrl = 'wss://vesslns.com/index';
const connect = new WebSocket(mainUrl);
```

Після успішного встановлення з'єднання викликається подія open. Організувати прослуховування цієї події можна, призначивши функцію зворотного виклику властивості onopen об'єкту connect, яка дозволяє викликати шар виконання бізнес-логіки для потрібної події, наприклад, ScriptService для опрацювання даних про сценарії.

Після відкриття WebSocket-з'єднання з сервером йому можна відправляти дані. Зробити це можна, наприклад в callback onopen.

Для отримання з сервера даних, відправлених з використанням протоколу WebSocket, можна призначити callback onmessage, який буде викликаний при отриманні події onmessage.

Для того щоб реалізувати WebSocket-сервер в середовищі Node.js, можна скористатися популярною бібліотекою ws. Ми застосуємо її для розробки сервера, але вона підходить і для створення клієнтів, і для організації взаємодії між двома серверами[20].

Розглянемо обробники WebSocket:

– обробник “setGrowRecord”, у якості даних передається масив етапів сценарію ScriptRecord;

– обробник “getGrowRecord” отримує масив етапів сценарію ScriptRecord;

– обробник “getDeviceState” отримує стан апаратного комплексу DeviceSnapshot;

– обробник “setFertilizersAmount”, у якості даних передається масив типу Double - значень об’єму ємкостей;

– обробник “getFertilizersAmount ” отримує масив типу Double - значень об’єму ємкостей;

– обробник “ checkClearWater ” отримує масив типу Number - значень рівня кислотності та засоленості;

– обробник “allowAutoWatering”, у якості даних передається Boolean змінна, яка дозволяє або не дозволяє автоматичний полив;

– обробник “hardReset” примусовий перезапуск апаратного комплексу;

– обробник “getHummidity” отримує змінну типу Number – значення рівня вологості;

– обробник “getStage” отримує змінну типу Number – значення поточного номеру етапу сценарію;

– обробник “getNotification” отримує змінну типу String – сповіщення про якусь подію в апаратному комплексі.

Опис REST API:

– POST /api/init – ініціалізація користувача при перезавантаженні сторінки;

– POST /api/login – ініціалізація користувача при перезавантаженні сторінки;

– GET /api/script/list – отримання списку сценаріїв;

– GET /api/script/${id}/profile – отримання деталей конкретного сценарію;

– POST /api/script/${id}/edit – редагування конкретного сценарію;

– GET /api/plant/list – отримання списку рослин;

– GET /api/plant/${id}/profile – отримання деталей конкретної рослини;

– POST /api/script/${id}/edit – редагування конкретної рослини;

– GET /api/user/list – отримання списку користувачів;

– GET /api/user/${id}/profile – отримання деталей конкретного користувача;

– POST /api/script/${id}/edit – редагування конкретного користувача;

– POST /api/device/stats – отримання статистики за певний період часу.

Реалізація обробки помилок.

Обернення екземпляру new WebSocket() на клієнтській стороні у блок try-catch для виявлення помилок типізації даних, передача callback (err) => handleError() у виклику методу send, бо виконується перевірка на null значення, цей же метод використовується на серверній стороні, додання методу onError(), бо це є специфікацією EventEmitter і дозволяє відстежувати помилки викликів.

Проблема черги повідомлень.

Щоб вирішити проблему великого потоку повідомлень з сервера на веб-клієнт, наприклад, в разі моніторингу великої мережі, необхідно якимось чином звести обробку кожного повідомлення до мінімальної кількості операцій, таким чином, зменшивши загальний час обробки повідомлень за певний проміжок часу, що дозволяє швидше звільняти ресурси для виконання інших завдань в однопоточному веб-клієнті. Буферизація даних дозволяє ефективно вирішити дану проблему. Така буферизація працює на зразок буферизації в процесорі, а саме при використанні буферизації веб-клієнт не оброблює жодних повідомленнь, які прийшли з сервера відразу, а складає їх в буфер, але не в масив, а зливає всі дані в одне повідомлення, щоб його можна було зручно в подальшому застосувати до наявних даних, і всього один раз за заданий проміжок часу[22].

Уявімо, що розроблюваний веб-додаток може містити як малу кількість даних в базі даних, так і дуже велику кількість даних. Для цього пропонується оновлювати таймер застосування даних мінімум до того моменту, коли жодне повідомлення не спаде за заданий раніше інтервал, а максимум до того моменту, коли значення суми тимчасових інтервалів (з огляду на поновлення таймерів) досягне якоїсь критичної позначки для форсованого поновлення. Наприклад, якщо задати мінімальний інтервал поновлення рівним 200 мс, то в тому випадку, якщо протягом 200 мс прийде тільки одне повідомлення, то дані з цього повідомлення застосуються до даних в уже існуючому веб-сховищі відразу після цього інтервалу, інакше, таймер оновиться і буде очікувати повідомлення, якщо повідомлення знову встигне прийти з сервера – таймер оновиться і так далі. Так буде відбуватися до тих пір, поки сума такого часу не досягне критичної позначки, яка дорівнює, наприклад, 1 секунді, і в цьому випадку дані, зліплені з всіх, які прийшли за 1 секунду повідомлень застосуються до вже існуючих даних в сховищі. Таким чином, при великому потоці даних оновлення даних у веб-інтерфейсі буде відбуватися не частіше, ніж 1 раз в секунду, а при малому потоці даних може відбуватися від 1 разу на 200 мс до 1 разу на 1 секунду.

В даній реалізації клас UpdateHandler відповідає за накопичення та застосування накопичених даних до сховища даних. У даній реалізації клас обробляє повідомлення про події, що сталися у великій мережі, тому, потрібна була

буферизація, щоб звести затримки операцій до мінімуму. Зовнішні обробники повідомлень WebSocket використовують метод bufferedUpdate, передаючи в нього повідомлення з оновленнями даних про події, що сталися в мережі.

3.3 Реалізація джерела даних

Для створення бази даних обрана СУБД MongoDB, бо дана СУБД має драйвер для веб-серверу, який написаний з використанням технології Node.js, а також у зв'язку з тим, що дані в MongoDB зберігаються в форматі JSON, що відповідає формату даних, що використовується в JavaScript[12]. Проаналізувавши принципи роботи системи, що розробляється, були встановлені наступні основні сутності для проектування БД:

– користувачі;

– пристрої;

– сценарії;

– поливи;

– параметри пристрою.

Колекція Users.

Ця колекція документів зберігає інформацію про зареєстрованих в системі користувачів. Структура колекції представлена в таблиці 3.1.

Таблиця 3.1 – Колекція Users

Назва поля	Тип даних
_id	ObjectID
full_name	String
created_at	Date
country_code	Number
role	String
user_devices	Array
token	String

Розглянемо значення кожного поля:

– _id – унікальний ідентифікатор;

– full_name – ім’я користувача для відображення у клієнтській частині;

– Created_at – дата створення акаунту користувача;

– country_code – код країни користувача;

– user_devices – масив апаратних пристроїв користувача;

– token – унікальний ключ авторизації.

Колекція Devices.

Ця колекція документів зберігає інформацію про апаратні комплекси. Структура колекції представлена в таблиці 3.2.

Таблиця 3.2 – Колекція Devices

Назва поля	Тип даних
_id	ObjectID
status	String
created_at	Date
active_script	Number
type	String

Розглянемо значення кожного поля:

– _id – унікальний ідентифікатор;

– status – статус пристрою;

– created_at – дата з’єднання апаратного комплексу користувача;

– active_script – ідентифікатор активного сценарію;

– type – тип пристрою.

Колекція Scripts.

Ця колекція документів зберігає інформацію про сценарії. Структура колекції представлена в таблиці 3.3

Таблиця 3.3 – Колекція Scripts

Назва поля	Тип даних
_id	ObjectID
hummidity	Number
temperature_out	Number
temperature_in	Number
ph	Number
ec	Number
p1	Number
p2	Number
p3	Number
p4	Number
p5	Number
up1	Number
up2	Number
up3	Number
up4	Number
up5	Number
created_at	Date

Розглянемо значення кожного поля:

– _id – унікальний ідентифікатор;

– hummidity – рівень вологості;

– temperature_out – температура навколишнього середовища;

– temperature_in – температура всередині апаратного комплексу;

– ph – рівень кислотності;

– ec – рівень засоленості;

– p1…p5 – об’єм основних контейнерів води та добрив на даний момент;

– up1…up5 – об’єм використаних контейнерів води та добрив;

– created_at – дата створення сценарію.

Колекція Waterings.

Ця колекція документів зберігає інформацію про сценарії. Структура колекції представлена в таблиці 3.4.

Таблиця 3.4 – Колекція Waterings

Назва поля	Тип даних
_id	ObjectID
hummidity	Number
temperature_out	Number
temperature_in	Number
ph	Number
ec	Number
usedp1	Number
usedp 2	Number
usedp 3	Number
usedp 4	Number
usedp 5	Number
water_amount	Number
active_script	String
created_at	Date

Розглянемо значення кожного поля:

– _id – унікальний ідентифікатор;

– hummidity – рівень вологості;

– temperature_out – температура навколишнього середовища;

– temperature_in – температура всередині апаратного комплексу;

– ph – цільвоий рівень кислотності;

– ec – цільовий рівень засоленості;

– usedp1…usedp5 – об’єм використаних води та добрив;

– active_script – назва сценарію;

– created_at – дата створення сценарію.

Враховуючи логічні зв’язки між елементами було розроблено схему бази даних, яка представлена у додатку Ж.

3.4 Реалізація серверного компоненту

Реалізації архітектури серверного додатку.

Для розробки веб-серверу було обрано технологію Node.js. Структура веб-серверу являє собою набір обробників HTTP-запитів і звернень до бази даних. Сервер надає призначеному для користувача інтерфейсу API для безпечної зміни або збереження даних в базі даних.

Розглянемо структуру Node.js застосування:

– app.js – точка входу у програму;

– api – endpoints Express у вигляді контролерів;

– config – налаштування додатку;

– loaders – модульний запуск процесів;

– models – моделі даних ;

– services – реалізація бізнес-логіки;

– subscribers – логіка обробки асинхронності.

Обрана трирівнева архітектура реалізується принципом розмежування обов’язків, розділяючи endpoints API, роботу з базою даних та бізнес-логіку.

Опис моделей Mongoose:

– UserModel модель користувачів;

– ScriptModel модель сценаріїв;

– WateringModel модель поливів;

– DeviceModel модель інформації про пристрій.

Реалізацію трирівневої архітектури серверного додатку на прикладі роботи з інформацією про стан апаратного комплексу представлено на рисунку 3.1.

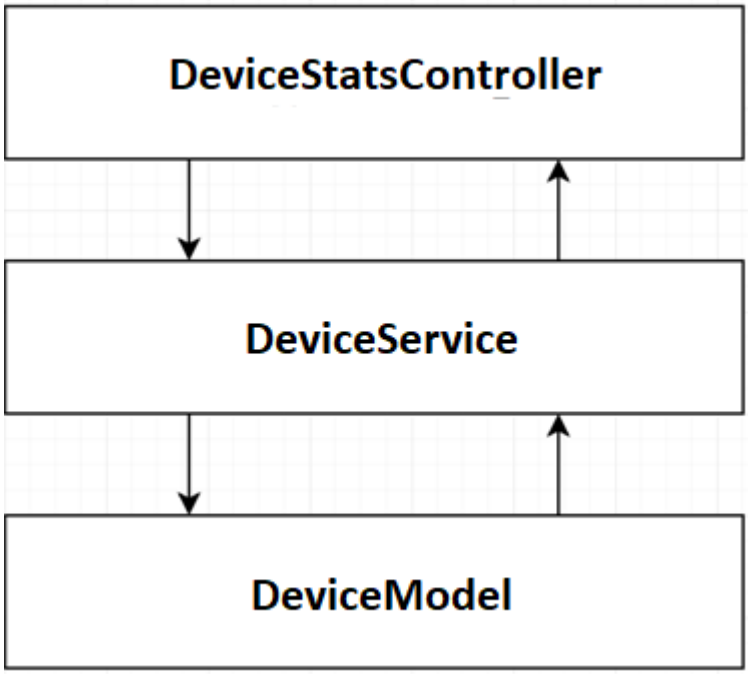

Рисунок 3.1 – Трирівева архітектура серверного компоненту

Опис контролерів Express:

– AuthController обробка логіки перевірки запитів та відправлення відповідей авторизації;

– UserController обробка логіки перевірки запитів та відправлення відповідей запитів користувачів;

– ScriptController обробка логіки перевірки запитів та відправлення відповідей запитів сценаріїв;

– WateringController обробка логіки перевірки запитів та відправлення відповідей запитів поливів;

– DeviceController обробка логіки перевірки запитів та відправлення відповідей запитів інформації про пристрій.

Опис сервісів:

– AuthService бізнес-логіка авторизації;

– SocketService бізнес-логіка WebSocket;

– SessionService обробка черги повідомлень;

– UserService бізнес-логіка роботи з користувачами;

– ScriptService бізнес-логіка сценаріїв;

– WateringService бізнес-логіка поливів;

– DeviceService бізнес-логіка інформації про пристрій.

Використання контролерів Express для реалізації бізнес-логіки не є вдалим рішенням через потенційні проблеми читабельності, масштабованості системи. Покривання код реалізації бізнес-логіки тестами стане більш складною задачею, бо доведеться проводити mock-тестування кожного окремого об’єкту запиту і відповіді.

Потрібно визначити, коли передавати дані на обробку в клас сервісу бізнес-логіки. Реалізація сервісних шарів вирішить проблему інкапсуляції бізнес-логіки, а клас роботи з даними при кожному новому виклику буде передавати дані у метод обробника логіки сервісного шару.

Це дозволяє відмовитися від імперативного способу, розділивши слухачів подів і методи обробки і повертати await-конструкції.

Використання паттерну Dependency Injection дозволяє докомпозувати код за допомогою передачі залежностей у відповідний клас або функцію[23].

Впровадження залежностей буде реалізовано за допомогою конструкторів класів. В класах Express використання Dependency Injection зводиться до створення екземпляру сервісу обробки бізнес-логіки та виклик потрібного методу, передавши в нього необхідну модель даних, після обробки даних відправляється відповідь на клієнтську частину.

Розглянемо використання Dependency Injection на прикладі UserService, де у контруктор класу передається екземпляр класу UserModel.

```
export class UsersService {
  constructor(
    private userModel: UserModel
  ) {}

  async getUsers() {
    return this.userModel.findAll();
  }
```

```
  async addUser(user) {
    await this.userModel.addUser(user);
  }

  async findUser(id) {
    return this.userModel.findUser (id);
  }
}

const usersService = new UsersService({
  userModel,
});
```

Рівень представлення інкапсулює логіку опрацювання REST Api.

Для реалізації відправки відповідей на запити клієнтів в Node.js є можливість описати маршрутизацію запитів за допомогою визначення відповідності шляху, по якому клієнт відправляє запит і функції обробника. Для цього в код головного модуля сервера було додано опис реалізації маршрутів для запитів авторизації, веб-додатків та синхронізації мобільного додатка.

В папці routes зберігаються файли в яких реалізовані маршрути відповіді додатку на клієнтські запити до конкретного URL. Кожен маршрут може мати одну або кілька функцій обробки, які виконуються при зіставленні маршруту.

Маршрути описано вище у розробці рівня комунікацій.

Структура визначення маршруту:

app.METHOD(PATH, HANDLER), де:

– app - є екземпляром express;

– METHOD - метод запиту HTTP;

– PATH - шлях на сервері;

– HANDLER - функція, виконувана при зіставленні маршруту.

У відповідності з кожним маршрутом порівняна модель, в якому описана реалізація. У вищевказаному REST API реалізовано усі операції CRUD (Create, Read, Update, Delete).

Для створеняя маршруту авторизації, потрібно використати наступні команди:

router.post(‗/auth‘, controller.auth);

router.post(‗/register‘, controller.register);

Інші маршрути створюються аналогічно.

Реалізація серверного компоненту аутентифікації користувачів.

Коли в системі створюється новий користувач, його пароль необхідно хешировати і зберегти в базі даних. Пароль в базі зберігають разом з адресою електронної пошти та іншими відомостями про користувача (наприклад, серед них може бути профіль користувача, час реєстрації і так далі), викликаючи фукнцію signUp(), схема авторизації розглянута у додатку И.

Розглянемо схему дій на рисунку 3.3, які виконуються в тому випадку, коли користувач намагається увійти в систему.

Дії, які відбувається при вході користувача в систему:

– Клієнт відправляє серверу комбінацію, що складається з публічного ідентифікатора і приватного ключа користувача, AuthService перевіряє чи має користувач право на доступ методом checkAuth();

– Сервер шукає користувача в базі даних, UserService.findUser();

– Якщо користувач існує в базі даних - сервер хеширує відправлений йому пароль і порівнює те, що вийшло, з хешем пароля, збереженим в базі даних, AuthService.checkPassword();

– Якщо перевірка виявляється успішною - сервер генерує токен JWT, AuthService.generateToken().

JWT - це тимчасовий ключ[25]. Клієнт повинен відправляти цей ключ сервера з кожним запитом до аутентифікації у кінцевій точці.

Дані токена можуть бути декодовані на стороні клієнта без використання секретного ключа або підпису.

Це може бути корисним для передачі, наприклад метаданих, закодованих всередині токена. Подібні метадані, можуть описувати роль користувача, його профіль, час дії токена, і так далі. Вони можуть бути призначені для використання у клієнтських-додатках.

Створення JWT буде реалізовано у функції generateToken(), яка потрібна нам для завершення роботи над сервісом аутентифікації користувачів.

Створювати JWT можна за допомогою бібліотеки jsonwebtoken. Ця бібліотека встановлена з npm.

Зі сторони клієнтської частин повинен відправлятися JWT в кожному запиті до захищеної кінцевій точці. JWT буде включено до заголовку запит Authorization.

На сервері, створюється клас, який представляє собою проміжне програмне забезпечення для маршрутів express.

Після реалізації цього механізму, маршрути зможуть отримувати відомості про користувача, який виконує запит.

3.5 Реалізація клієнтської частини

Для розробки клієнтської частини було обрано фреймворк Angular.

Проаналізувавши функціональні вимоги веб-застосування, було створено наступну побудову компонентів для реалізації усієї потрібної бізнес-логіки проекту. Діаграма компонентів Angular представлена у додатку К.

Компонентна структура.

Система поділяється на дві секції – Admin Area і Client Area, доступ до останнньої є лише у адміністраторів, в ній можна безпосередньо втручатися у процес роботи користувацьких апаратних комплексів, додавати нових користувачів, створювати сценарії.

У клієнтській частині користувач може під’єднати свій апаратний комплекс до системи, додати рослину, вибрати або ж створити свій власний сценарій розвитку, відстежити прогрес зростання, та параметри системи.

Спільними модулями для обох секцій є секції створення та редагування сценаріїв, так як і перегляд інформації про пристрої.

Спільними для ClientAreaModule і AdminAreaModule є компоненти:

– ScriptList і ScriptManage для перегладу там можливості корегування сценаріїв розвитку;

– DeviceSideStats і DeviceDetails інформація у одному із сайдбарів системи про поточний стан системи.

Окремим модулем ClientAreaModule є PlantsList і PlantsManage для перегляду і редагування сценаріїв розвитку поточних рослин користувача.

Окремими модулями AdminAreaModule є UserList та UserManage для перегляду і редагування користувачів системи.

Загальні модулі системи:

– AppModule – точка входу у клієнтське застосування Angular;

– DataModule – інкапсуляція зв’язку з сервером;

– AuthModule – логіка аутентифікація користувача;

– TestModule – логіка тестування;

– ToastsModule – модуль сповіщень;

– LocalStorageModule – логіка роботи з локальним сховищем;

– ThemeModule – реалізація візуальних тем додатку;

– SidebarModule – інформація про користувача;

– NotificationsModule – модуль повідомлень про стан системи.

Детальна схема компонентів клієнтської частини розглянута у додатку К.

Клієнтська аутентифікація.

У системі є дві ролі користувачів – Адміністратор і власне Користувач, для реалізації прав доступу використовуються технологія JWT.

Коли користувач входить на будь-яку веб-сторінку зі своїм ім'ям користувача і паролем, сервер аутентифікації зазвичай створює і відправляє назад JWT. Потім цей JWT передається разом з наступними викликами API на сервер. JWT залишається в силі, поки не закінчиться термін його дії або користувач не вийде з програми.

LoginComponent – перше, що бачить користувач у системі, вводить email і пароль та відправляє запит на аутентифікацію. ReactiveForms виконують валідацію форми, і в разі успішного проходження передають дані користувача до AuthService, де відбувається процес запиту авторизації. Якщо авторизацію виконано успішно, клієнт отримує об’єкт у якому описана вся інформація про користувача, токен, його пристрої, рослини, налаштування, власноручно створені сценарії. Токен зберігається

у локальному сховищі для можливості перезавантаження сторінки без потреби проходити процедуру аутентифікації заново. Усі подальші запити до серверу включають у собі токен авторизації, який, за допомогою AuthInterceptor передається до заголовку запиту Authorization.

Послідовність роботі реалізації аутентифікації на клієнтській частині показано на рисунку 3.2.

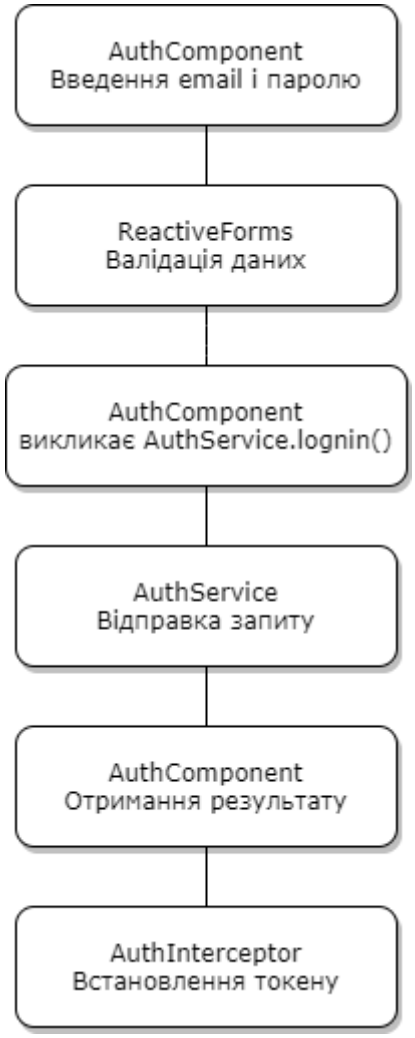

Рисунок 3.2 – Механізм клієнтської аутентифікації

Керування станом додатку.

Angular робить можливим використовувати інтерфейс користувача без перезавантаження сторінки. Лише певні компоненти будуть перезавантажені. Якщо механізм ChangeDetection знайде відмінності між попереднім і поточним значенням

змінної, він зможе виконати повторний рендеринг сторінки[24]. Дана ситуація виникає при асинхронному отримання даних з серверу.

Розглянемо керування станом додатку на прикладі взаємодії з сервером за допомогою WebSocket. Обробка повідомлень є асинхронною, бо не блокує потік виконання. Головним механізмом роботи з асинхронністю у Angular є Observable, який є потоком подій, реалізуючи паттерн Observer.

Будь-який HTTP запит буде повертати об’єкт Observable, а компонент, який виконує запит, викликає функцію subscribe() та отримує дані, передані від сервісу. Також Observable створюється для реалізації будь-якої асинхронної операції, наприклад, при запиті інформації про поточний стан системи, яка реалізована у вигляді статистики, що постійно обновлюється,

ConnectorService створює Subject (різновид спостережуваного об’єкту RxJS, якому можна як надсилати повідомлення, так і підписуватися на його оновлення за допомогою перетворення його у Observable, this.messageSubject.asObservable()).

При ініціалізації компоненту DeviceDetails у базовому методі класу Component – ngOnInit, який виконується один раз при ініціалізації компоненту, буде створена підписка на повідомлення про зміну рівня вологості пристрою.

Приклад отримання поточного рівня вологості:

```
ngOnInit() {
        this.connector.getResponse(‘getHummidity’).subscribe(res => {
                this.currentHummidity = res.data.hummidity;
        })
}
```

3.6 Користувацький інтерфейс

Для створення інтуітивно зрозумілого UI/UX дизайну потрібно поліпшувати користувацький досвід за допомогою вже впізнаваних елементів інтерфейсу[27]. За допомогою цих елементів клієнтська частина виглядає структурованою, приємною для користування та легкою для розуміння.

У даній системі використано фреймворк ng-bootstrap, який використовує компоненти стилів Bootstrap 4 і адаптує їх до використання у середовищі Angular[26].

Розглянемо використані компоненти інтерфейсу:

– Ngb-accordion для можливості приховання повної інформації про сценарій у модулі редагування;

– Ngb-alert реалізує компонент сповіщень, наприклад, про відповіді серверу;

– Ngb-dropdown для можливості вибору опцій форми;

– Ngb-modal для швидкого створення модальних вікон, які легко кастомізуються;

– Ngb-pagination для реалізації сторінок у компонентах, де користувач переглядає списки;

– Ngb-table для швидкого створення таблиць;

– Ngb-timepicker для створення опцій вибору дати та часу;

– Ngb-toast – повідомлення про стан з’єднання.

Основна стилізація додатку виконана у стилі Material design, який відомий у всьому світі за рахунок свого простого користувацького інтерфейсу, плавних анімацій та зрозумілих функціональних елементах.

3.7 Тестування веб-додатку

Для покращення користувацького головним чинником є час завантаження додатку. Якщо сторінка завантажується більш ніж 3 секунди, багато користувачів просто закриють вкладку з додатком, а у мережах, де швидкість з’єднання достатньо мала, це викликає значні проблеми при користуванні додатку та нестабільну роботу.

У якості сервісу тестування швидкості завантаження додатку було використано Google PageSpeedInsights.

Результати тестування продемонстровано на рисунку 3.3.

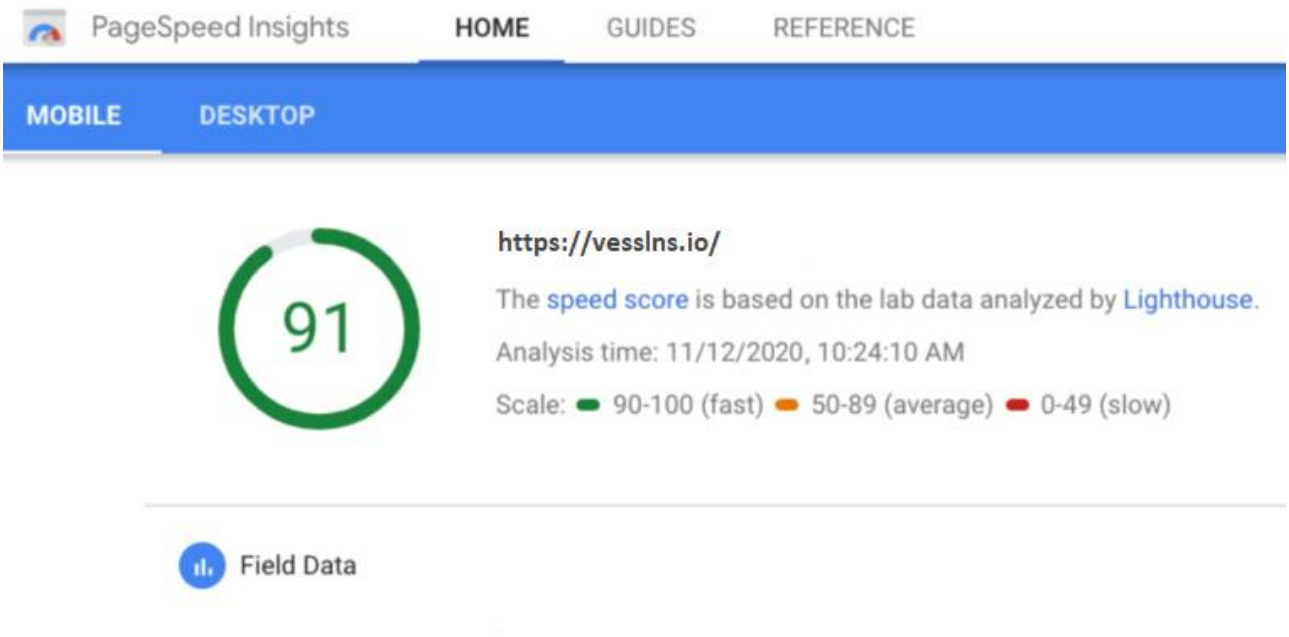

Рисунок 3.3 – Результати Google PageSpeedInsights

3.8 Експериментальні дослідження

Додаток, що розроблюється у системі має відповідати наступним вимогам:

– авторизація користувача;

– під’єднання апаратного комплексу до системи;

– додавання нової рослини у систему;

– створення або вибір сценарію розвитку рослини;

– редагування сценарію розвитку;

– відслідковування змін у пристрої;

– перегляд статистики за обраний період часу.

Використання системи починається зі сторінки авторизації, де користувач має ввести свої особисті дані – email і пароль. Це зображено на рисунку 3.4.

Після успішної авторизації користувач потрапляє на сторінку під’єднання апаратного комплексу до системи, детальніше на рисунку 3.5.

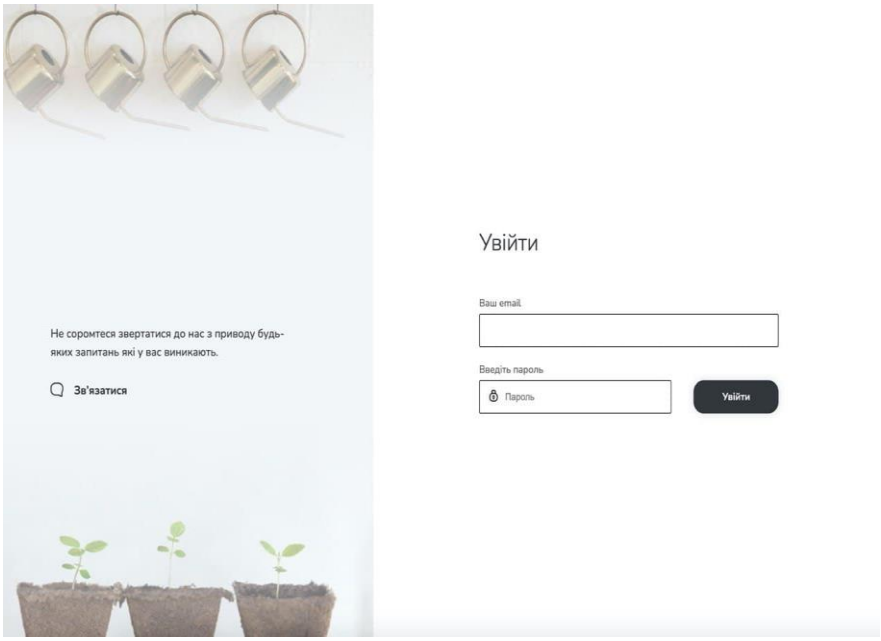

Рисунок 3.4 – Сторінка авторизації

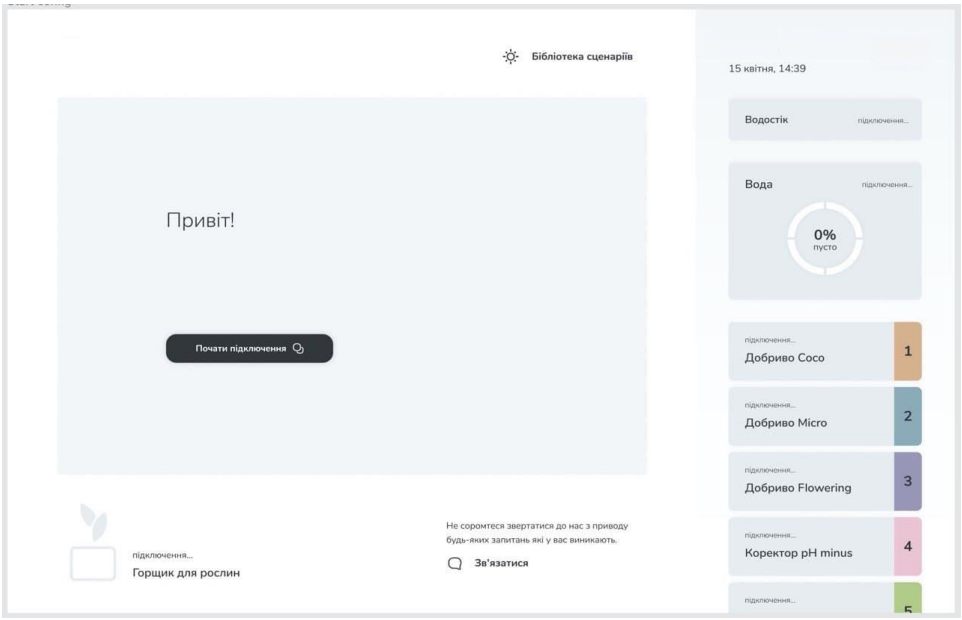

Рисунок 3.5 – Під’єднання апаратного комплексу

На даному етапі розпочинається під’єднання до системи, завантаження усіх необхідних добрив, під’єднання пристрою до мережі та калібрування апаратного комплексу. Після успішного проходження даного етапу, користувач потрапляє на головну сторінку, рисунок 3.6.

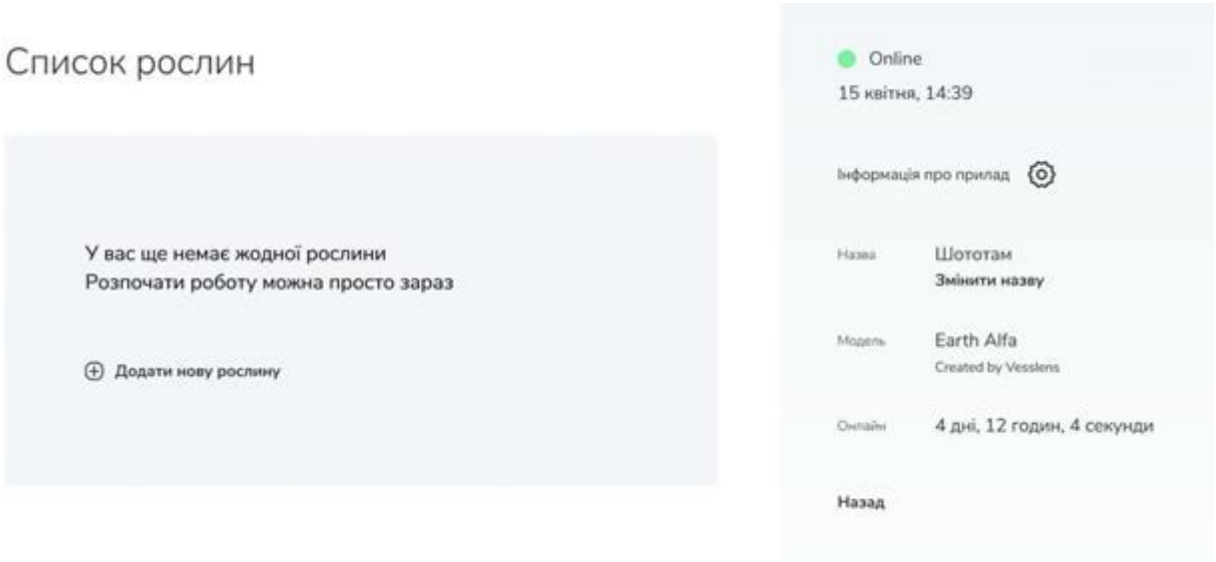

Рисунок 3.6 – Головна сторінка

На головній сторінці користувач може бачити основну інформацію про систему.

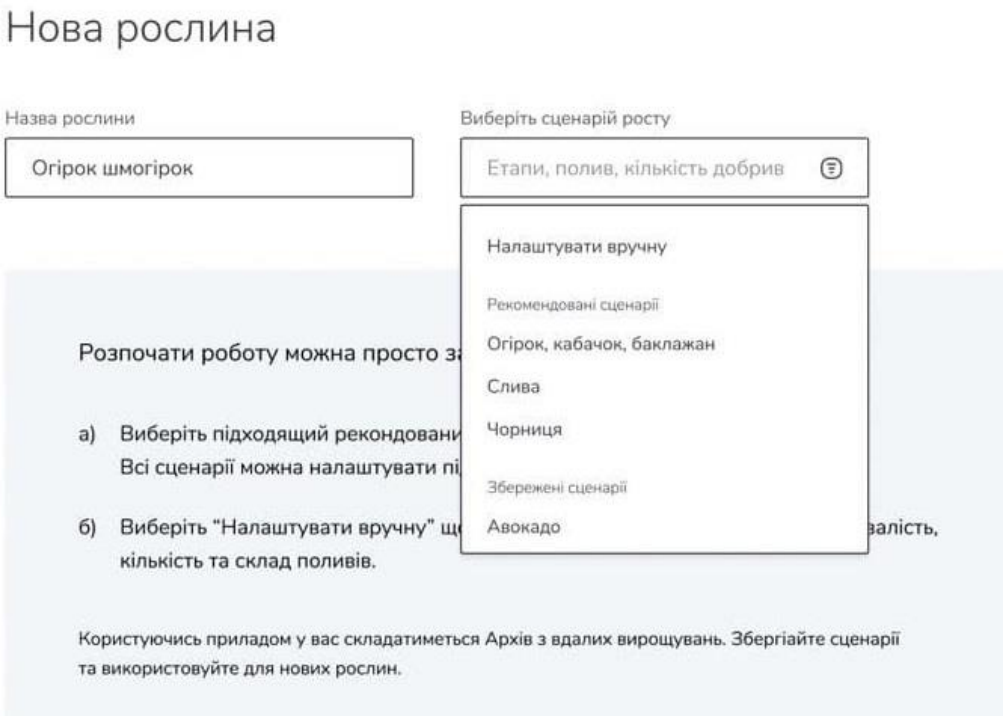

Рисунок 3.7 – Додавання нової рослини

На головній сторінці знаходиться кнопка для додавання нової рослини та інформація про стан пристрою, щоб розпочати зрощення власної рослини, потрібно ввести назву і обрати сценарій, рисунок 3.7.

Після вводу назви рослини є два варіанта початку роботи – вибрати рекомендований сценарій, який вже було створено, або створити свій власний сценарій. Зліва користувач може побачити навігаційне меню, завдяки чому він має можливість перейти до бібліотеки сценаріїв або архіву зрощень.

На рисунку 3.8 зображена схема розробки визначених етапів зрощення рослини. Користувач може обрати будь-яку кількість етапів сценарію, побачити коротку інформацію про кожен з них та редагувати при бажанні.

Користувач має можливість налаштувати усі необхідні йому показники зрощення рослини, а саме:

– назва етапу;

– тривалість етапу;

– вологість;

– рівень pH та EC;

– склад поливу.

Система буде працювати для наближення реальних показників до ідеального значення, яке вказано у сценарії. Кожен етап має свою назву, тривалість, параметри та склад поливу, який вказує користувач

Детальну статистику поливів можна побачити на рисунку 3.9.

Налаштування сценаріїв є основною функцією системи керування, реалізуючи усі функціональні вимоги.

Після налаштування сценаріїв користувач може запустити апаратний комплекс і розпочати зрощення рослини.

Слід зазначити, що користувач може змінювати параметри налаштувань у системі під час протікання сценарію, апаратний комплекс отримає оновлені дані, а тривалість процесу зрощення буде перерахована.

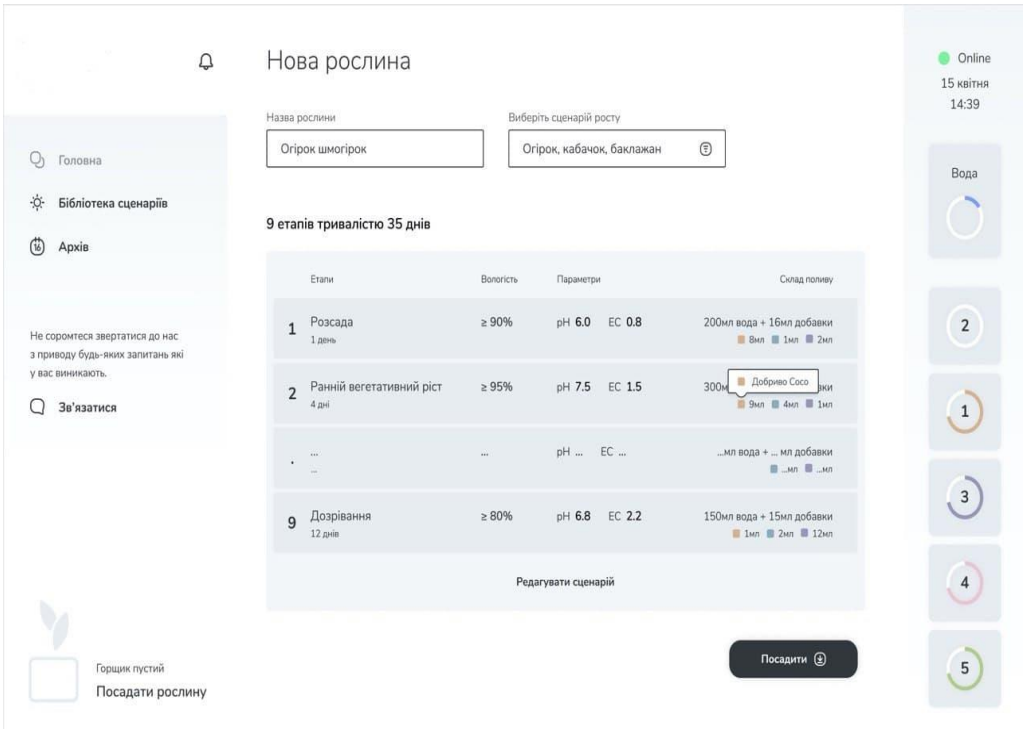

Рисунок 3.8 – Розробка етапів зрощення

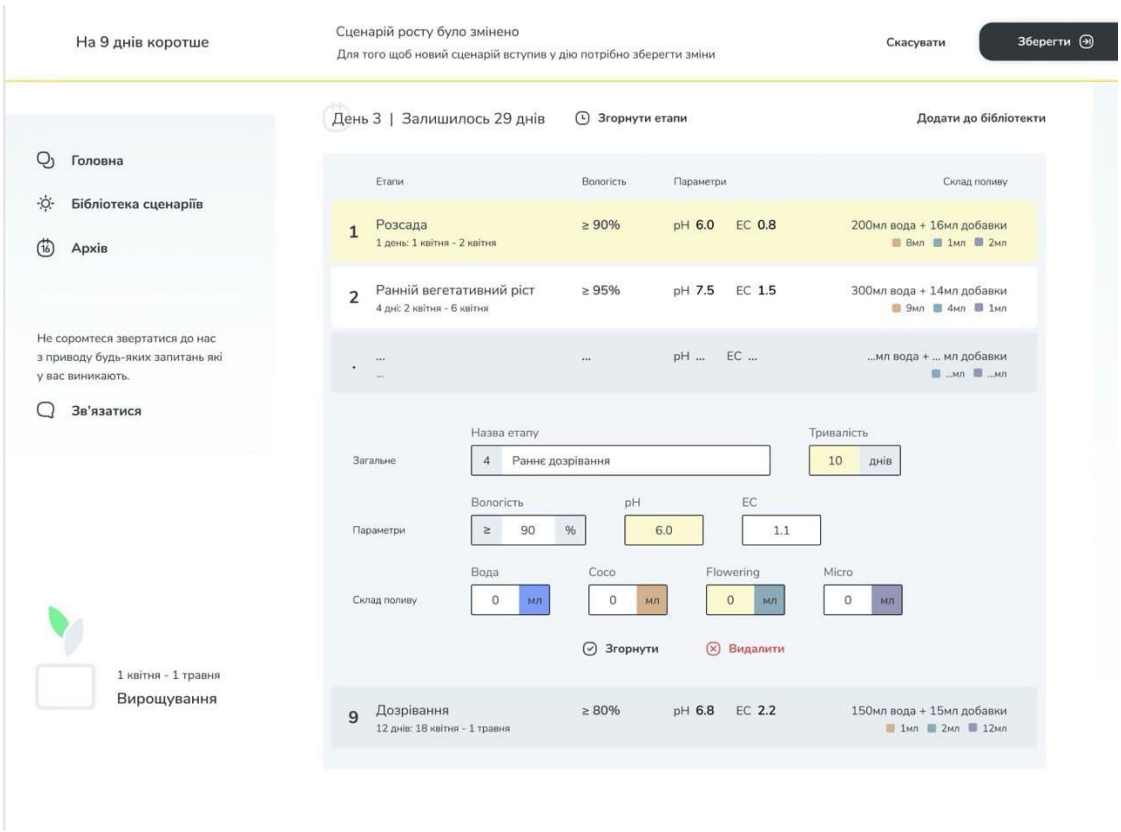

Рисунок 3.9 – Детальне редагування етапу сценарію розвитку

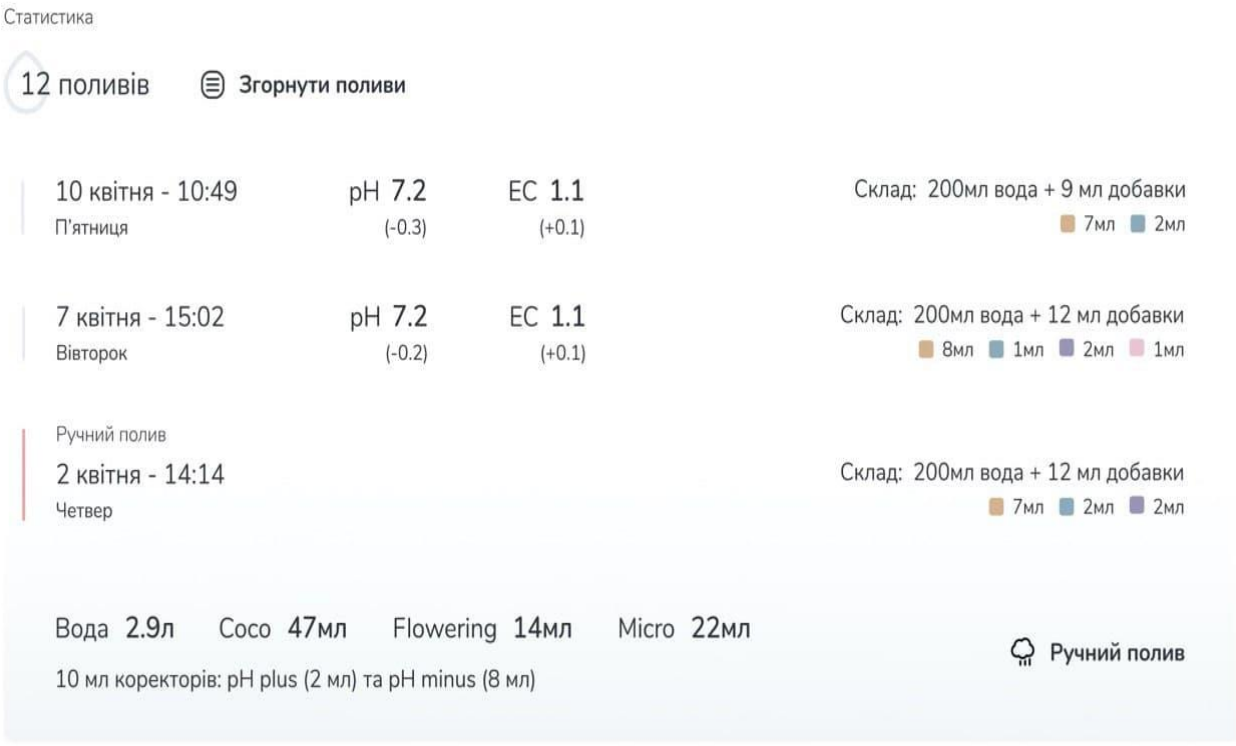

Рисунок 3.10 – Статистика поливів пристрою

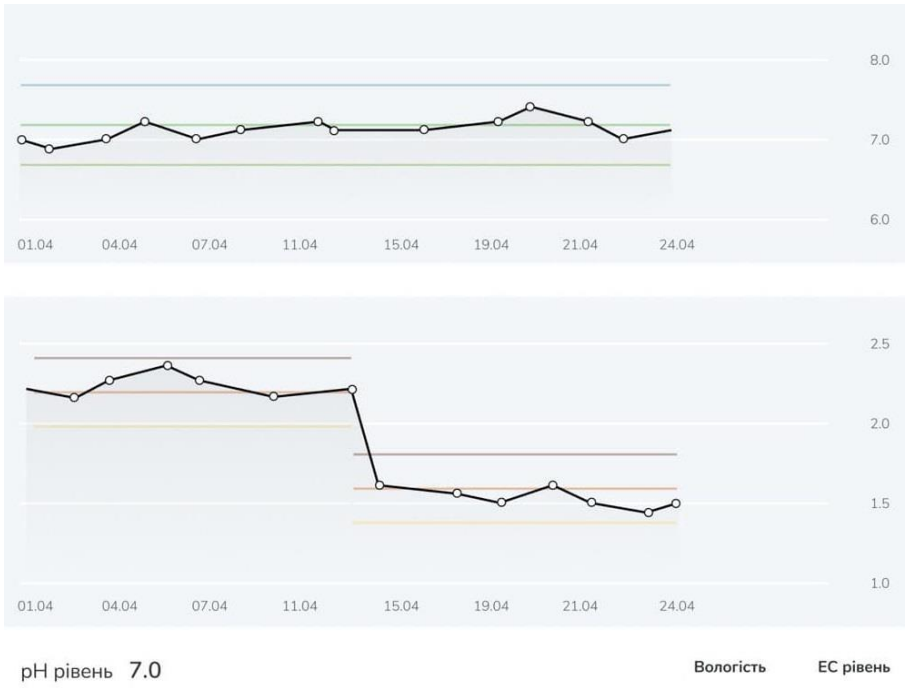

Рисунок 3.11 – Графіки засоленості і кислотності

Користувач має змогу відслідковувати статистику поливів, бачити показники параметрів, які були зафіксовані у той момент часу, у дужках пишеться відхилення від норми, яке вказано у визначеному етапі сценарію розвитку рослини, показано стан апаратного комплексу на поточний момент, склад кожного поливу з детальною інформацією про об’єм добрив рисунок 3.10.

На графіках засоленості рисунку 3.11 і кислотності користувач може відслідкувати як само змінювалися показники, за який проміжок часу система наближували показники до тих, що вказані у сценарії розвитку.

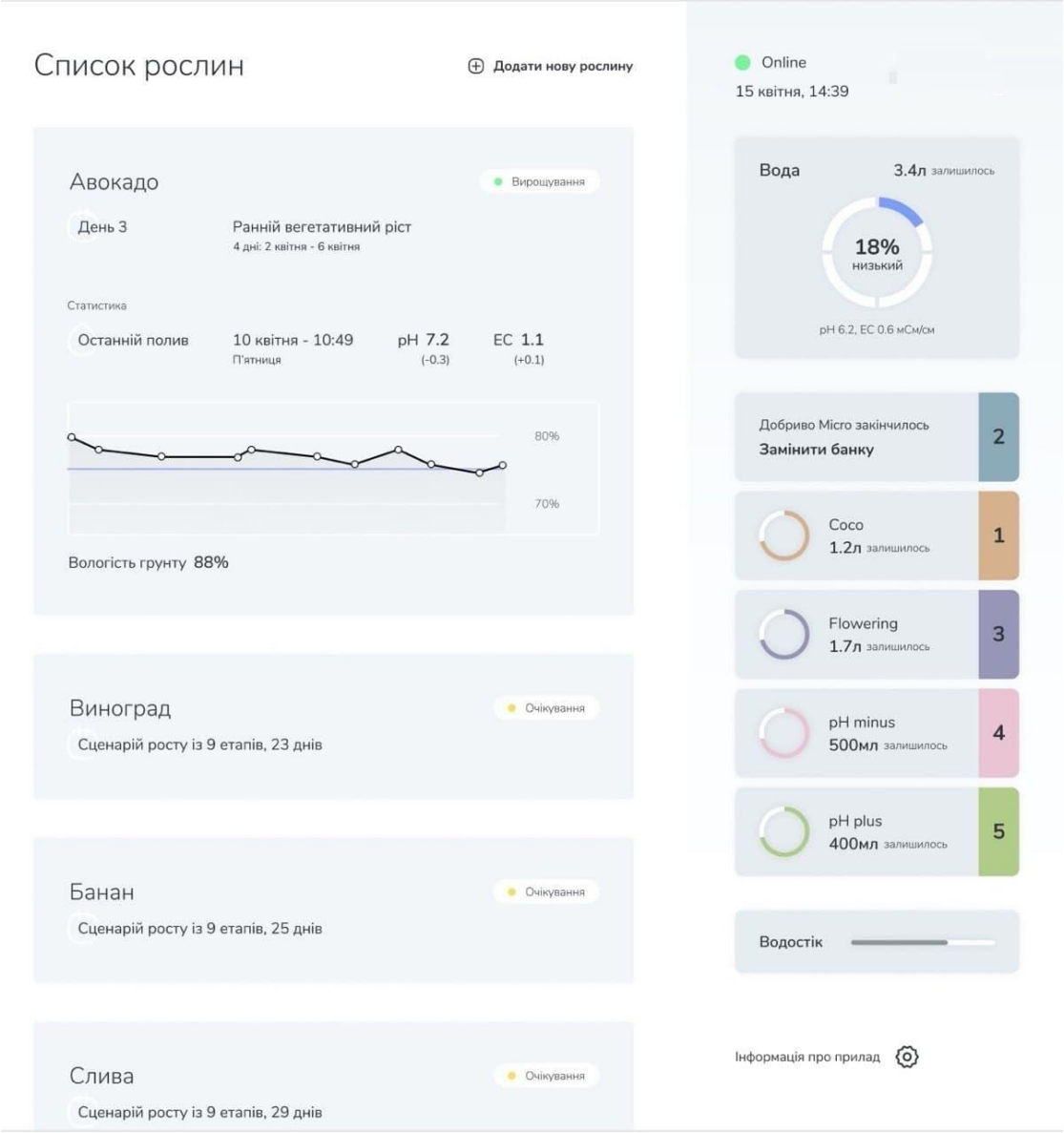

Рисунок 3.12 – Список рослин з параметрами у реальному часі

На рисунку 3.12 зображено список рослин, які користувач планує виростити, на даний час він бачить актуальну інформацію про стан своєї рослини. Щоб відслідковувати події, для підвищення користувацького досвіду вони були поділені по кольорам які відбуваються всередині можна скористатися сповіщеннями, рисунок 3.13.

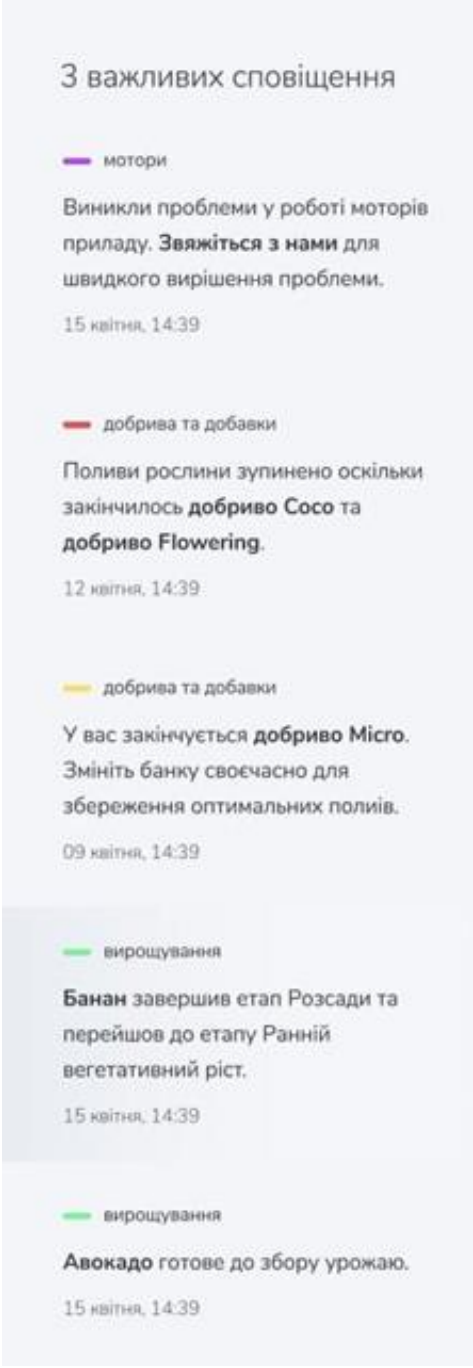

Рисунок 3.13 – Компонент повідомлень про події всередині пристрою

У рамках використання системи можуть виникати ситуації, які будуть потребувати безпосереднього втручання адміністратора системи, бо це зменшує ризики виведення системи із робочого стану та надає технічну підтримку користувачам. Це здійснюється за допомогою Адміністративної панелі, де власне адміністратор обирає необхідного йому користувача і під’єднується до його системи з метою якомога швидшого вирішення проблем чи виправлення критичних помилок

На рисунку 3.14 зображено прецедент адміністратора, який може налаштувати конкретний пристрій або виконати процедуру скидання усіх основних налаштувань системи.

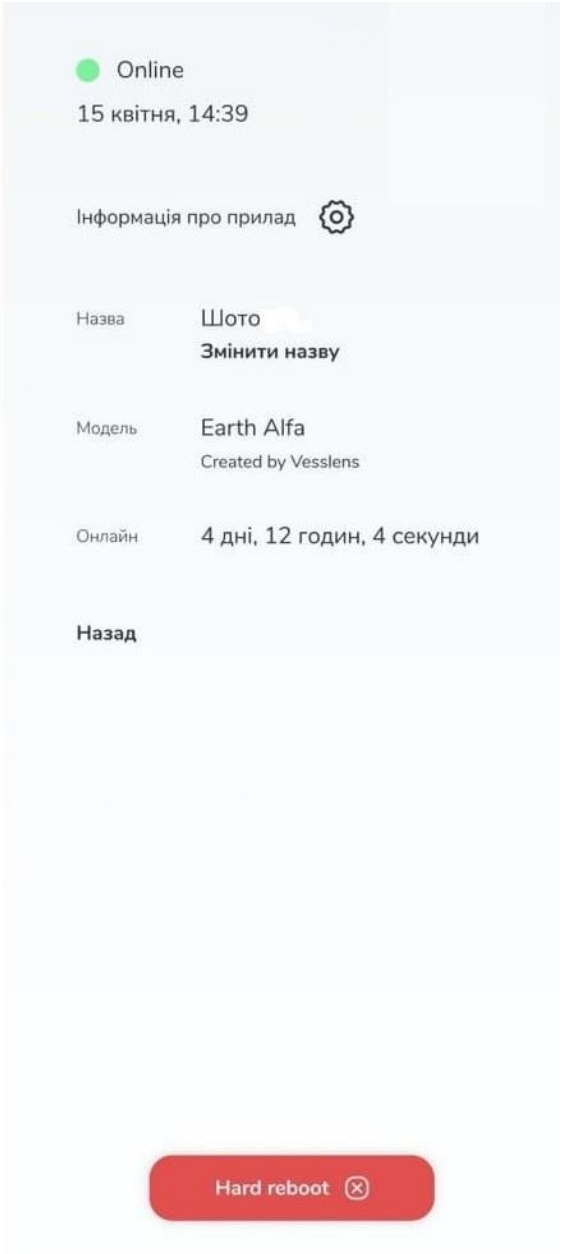

Рисунок 3.14 – Можливість повного перезавантаження системи

3.9 Висновки

В даному розділі було наведено інженерне рішення по реалізації декомпозиції поточної архітектури та її модулів. Серед яких сховище даних, що представлено NOSQL базою даних MongoDB, з відповідною схемою даних, що відповідає поставленим вимогам до моделей, що зберігатимуться, серверний компонент що реалізований на технології Node.js та його декомпозицією в вигляді модулів авторизації, з’єднання з апаратним комплексом у режимі реального часу, реалізації бізнес-логіки системи, клієнтський додаток що реалізований на технології Angular та його декомпозицією в вигляді модулів авторизації, маршрутизації, з’єднання у режимі реального часу, реалізації бізнес-логіки системи. Наведено експериментальні дослідження, продемонстровано роботу продукту, наявні функціональні можливості, що доступні для кожного користувача системи. Система реалізує весь функціонал відповідно до вказаних вимог в межах аналізу предметної області. Програмний продукт протестований в повному обсязі за допомогою мануального тестування набором користувачів

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

Система керування комплексом підтримки життєдіяльності рослин, та подальша її модернізація була взята за основу для ідеї створення стартап-проекту. Опис ідеї описано у таблиці 4.1.

Таблиця 4.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система управління комплексом підтримки життєдіяльності рослин	Використання приватними особами, малим та середнім бізнесом у напрямку автоматизованих агрономічних систем	1. Ретельне налаштування сценаріїв розвитку рослин 2. Автоматизація процесів підтримки життєдіяльності рослин 3. Віддалений контроль за допомогою додатку

Як можна зазначити, система може бути застосована малим та середнім сегментом агрономічного бізнесу. Потрібно визначити основні характеристики системи та порівняти їх з аналогами - GroLab та Intelligrow (див. таблицю 4.2).

Таблиця 4.2 – Характеристики ідеї проекту

					W	N	S
				Intelligrow			
.	Вартість	10000 грн	80000 грн	56000 грн	–	–	+
.	Вартість обслуговування	1500 грн/місяць	5 000 грн/місяць	4 000 грн/місяць	–	–	+
.	Кросбраузерність/Кросплатформеність	Google Chrome 60.0+, Mozilla Firefox 51.0+, Safari6+, IE11+, Opera11+	Google Chrome 60.0+, Mozilla Firefox 51.0+, Safari6+, IE11+, Opera11+	Google Chrome 60.0+, Mozilla Firefox 51.0+, Safari6+, IE11+, Opera11+	–	+	–

.	Інтеграція із смартфоном	iOS6.0+ Android5	iOS7.0+ Android6	iOS7.1+ Android6	–	–	+
	Робота в мережі	Присутня	Присутня	Присутня	–	+	–
	Робота в локальній комп’ютерній мережі	Присутнє	Відсутнє	Присутнє	–	–	+
	Час завантаження	0.4 с	0.5 с	0.7 с	–	–	+
	Аналіз сценаріїв розвитку	Присутній	Відсутній	Відсутній	–	–	+
	Вірогідність відмови	Низька	Середня	Висока	–	–	+

	Стійкість до відмов	Середня	Висока	Висока	–	+	–
	Наявність системи резервного копіювання	Присутня	Присутня	Присутня	–	+	–
	Відновлюваність	Висока	Висока	Висока	–	+	–
	Технологічна собівартість	8 000 грн	20000 грн	28 000 грн	–	+	–
	Простота користування	Висока	Середня	Середня	–	–	+
	Наявність технічної документації	Присутня	Присутня	Присутня	–	+	–
	Автоматична обробка заявки на ремонт	Присутня	Присутня	Присутня	–	+	–
	Наявність служби технічної підтримки	Присутня	Присутня	Присутня	–	+	–

	Автоматична система оплати сервісу	Присутня	Присутня	Присутня	–	+	–
	Надання інтерфейсу для зовнішнього програмного забезпечення	Відсутнє	Присутнє	Відсутнє	+	–	–
	Інтеграція з платформами и інших сервісів	Присутнє	Частково присутнє	Відсутнє	–	–	+
	Підтримка багатьох апаратних комплексів	Присутнє	Відсутнє	Відсутнє	–	–	+

Аналізуючи систему, зрозуміло, що присутні можливості до масштабування, помірну швидкість роботи. Недолікоми є відсутність інтерфейсів для зовшнього програмного забезпечення, але це не має великого впливу на продукт.

4.2 Технологічний аудит ідеї проекту

Технічний аудит ідеї важливий для визначення необхідних технологій та їхніх доступностей для реалізації проекту. Результат аудиту наведено у таблиці 4.3.

Таблиця 4.3 - Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її	Наявність	Доступність
1		Angular	Наявні	Доступні
2		React	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: Angular				
4		REST	Наявні	Доступні
5		XML	Наявні	Доступні
6		SOAP	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: REST				

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Наступним етапом є вивчення ринку та визначення загроз для планування напрямів розвитку проекту, врахувуючи конкуренцію та потреби потенційних клієнті. Аналіз потенційного ринку показано в таблиці 4.4.

Таблиця 4.4 - Попередня характеристика потенційного ринку проекту

п/п	Показники стану ринку (найменування)	Характеристика
	Кількість головних гравців, од	3
	Динаміка ринку (якісна оцінка)	Зростає
	Наявність обмежень для входу (вказати характер обмежень)	Недискримінаційні якісні
	Специфічні вимоги до стандартизації та сертифікації	Відсутні
	Середня норма рентабельності в галузі (або по ринку), %	63,5%

Провівши аналіз потенційного ринку визначено, що ринок має усі необхідні якості, щоб бути привабливим для інвестицій. В таблиці 4.5 наведено характеристики потенційних клієнтів стартап-проекту.

Проаналізувавши характеристику потенційних клієнтів зрозуміло, що цільовою аудиторією системи є малий та середній бізнес у агрономічній сфері. Але різницею є масштаби виробництва та різний рівень вимог до функціональності системи.

Далі потрібно проаналізувати ринкову середовище, а саме, фактори загроз (таблиця 4.6) та можливостей (таблиця 4.7), які впливають на впровадження проекту.

Таблиця 4.5 - Характеристика потенційних клієнтів стартап-проекту

		Цільова	Відмінності у	
1	Бажання автоматизувати процеси зрощення та підвищити економічну ефективність виробництва	1. Малий та середній бізнес у агрономічній галузі	1. Масштаби виробництва 2. Різний рівень вимог до функціональності системи	1. Простий інтерфейс користувача 2. Можливість детального налаштування сценаріїв роботи

Таблиця 4.6 - Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція
1	Крадіжка інтелектуальної власності	Крадіжка інноваційної ідеї	Відсудження прав інтелектуальної власності Захист інтелектуальних технологій Обмежений доступ до цінної інформації для співробітників

2	Мала кількість стартових капіталовкладень	Неможливість створення базової версії продукту через недостатню кількість інвестицій	Пошук нових можливостей для отримання інвестицій

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція
1	Отримання необхідних інвестицій	Збір команди, початок розробки та реалізації мінімального за функціоналом продукту	Розробка продукту
2	Співпраця з малим бізнесом у агрономічній сфері	Малий бізнес може протестувати бета-версію продукту, дати відгуки, після яких буде можливість удосконалити продукт та приділити увагу слабким місцям	Тестування на реальних користувачах Маркетинг

3	Підвищений інтерес користувачів	Агрономічна сфера є ключовою в багатьох економіках світу	Масштабування виробництва та пошук нових ринків збуту
4	Маркетингова політика та таргетна реклама	Підвищення користувацького інтересу через рекламу, орієнтовану на цільові групи користувачів	Покращення стабільної роботи системи, модернізація та масштабування системи Розробка міні-версій системи Розширення функціоналу для ще більш швидкого залучення нових користувачів
5	Декоративна сфера	Використання систем у громадських місцях у якості екологічної реклами	Приділення уваги дизайну якомога ефектному Виду продукту

Аналіз особливостей конкурентного середовища наведено у таблиці 4.8.

Таблиця 4.8 - Ступеневий аналіз конкуренції на ринку

Особливості		Вплив на діяльність
Олігополія	Незначна кількість конкурентів Велика ринкова сила Схожість використовуваних технологій	Інформування ринку щодо появи нової веб-служби Співпраця із провідними сервісами
Галузевий	Загроза появи нових конкурентів Ринкова влада споживачів Висока потреба у товарі	Інформування ринку щодо якості використовуваної новаторської технології Пропозиція гнучких цін
Внутрішньогалузева	Діяльність в одній галузі економіки Надання сервісів одного типу	Зменшення вартості сервісу Примноження каналів розподілу
Товарно-видова	Надання різних сервісів одного виду	Маркетингова політика
Цінова	Використання цін для покращення економічних умов збуту	Зменшення вартості сервісу Використання нових каналів розподілу

Марочна	Пропозиція схожого сервісу Спільна цільова аудиторія	Інформування ринку щодо якості використовуваної новаторської технології Примноження каналів розподілу

Надалі треба визначити 5 основних факторів, які мають вплив на привабливість вибору ринку, оглядаючи характер конкуренції (таблиця 4.9.)

Таблиця 4.9 – Аналіз конкуренці в галузі за М. Портером

	Прямі	Потенційні	Постача		Товари-
	мій продукт,	Розмір капіталовкладень, Витрати на масштабуванні		– Змінні витрати: Виробничі непрямі дегресивні – Системи інформації: чутливості до цін:	Копіювання функціоналу, юторів, Мініміза ція цін

	Прямі	Потенційні	Постача		Товари-
				на цінність продукту Продуктова відмінність: якість, легкість в освоєнні функціоналу, ретельну обслуговування Методи контролю якості: тестування реальними користувачами, модульне тестування	

	Прямі	Потенційні	Постача		Товари-
Висновки	CR4 = 94% Індекс Херфіндаля-Хіршмана (HHI) = 7225 Значення показників вказує на (монополізацію) даного ринку	Вхід на Ринок буде забезпечено мінімізацією цін, швидкість та якість наданих послуг споживачам і співпраця із основними учасниками ринку. В результаті ретельного аналізу продуктів в агрономічній сфері потенційних конкурентів немає	Відсутні	Клієнти ставлять умови цінової політики, надавання високої і якості послуг.	Пропозиція вигідних умов дистриб'юторам, забезпечення захисту авторських прав і, гнучка цінова політика

Провівши аналіз та врахувавши характеристики проекту, зробимо порівняльний аналіз сильних та слабких сторін проекту (таблиця 4.10) та опишемо таблицю можливостей впровадження проекту, яка являє собою SWOT-аналіз (таблиця 4.11).

Таблиця 4.10 - Порівняльний аналіз сильних та слабких сторін проекту

			Рейтинг товарів-конкурентів у порівнянні						
			–3	–2	–1	0	+1	+2	+3
1	Унікальність сервісу	15						+	
2	Модель бізнес для бізнесу	18							+
3	Цінова політика	10							+
4	Додаткові послуги	18					+		

Таблиця 4.11 - SWOT- аналіз стартап-проекту

Сильні сторони: Якість та іноваційність Маленька ціна Широкий вибір версій	Слабкі сторони: Малий стартовий капітал Бізнес модель залежить від політики Необхідність широкого функціоналу на старті
Можливості: Інвестиції Реалізація бізнес-плану Висока зацікавленість цільової аудиторії	Загрози: Крадіжка інтелектуальної власності Недостатня реклама Недовіра користувачів

Надалі треба розробити альтернативу ринкового впровадження стартапів та визначити найкращий час для їхньої ринкової реалізації, звертаючи увагу на системи конкурентів, що можуть бути виведені на ринок

Аналіз альтернатив реалізується за допомгою строків та та ймовірності отримання ресурсів (таблиця 4.12).

Таблиця 4.12 - Альтернативи ринкового впровадження стартапів

	Альтернатива	Ймовірність	
1	Створення власної компанії	Йовірне	10 місяців
2	Маркетингова кампанія	Ймовірне	3 місяці
3	Пропонування безкоштовних сервісів	Ймовірне	3 місяці
4	Пошук бізнесів іншої галузі для співпраці	Малоймовірне	6 місяців
Обрана альтернатива: Створення власної компанії			

4.4 Розробка ринкової стратегії проекту

Таблиця 4.13 Вибір цільових груп потенційних споживачів

	Обрана	Стратегія	конкурентоспромож альтернативи	Базова
	Надання продукту	Вибірковий		Стратегія

	постачальникам продукції у якості веб-додатку	розподіл	Ефективна реклама Низькі витрати	диференціації

Стратегія охоплнення ринку розглянута у таблиці 4.13 Проаналізувавши цільові групи користувачів треба визначити стратегію розвитку (таблиці 4.14).

Таблиця 4.14 - Визначення базової стратегії розвитку

		Готовність	Орієнтовний		
1	Приватні особи	Висока	81%		Високі бар’єри входу
2	Малий бізнес	Середня	70%		Високі бар’єри входу
3	Середній бізнес	Висока	78%		Низьки бар’єри входу
Які цільові групи обрано: Малий бізнес та приватні особи					

Наступим етапом є визначення стратегії конкурентної поведінки (таблиця 4.15).

Таблиця 4.15 - Визначення стратегії конкурентної поведінки

		Чи буде компанія шукати існуючих у конкурентів?	Чи буде компанія копіювати конкурента, і які?	
1	Ні	Так та залучати нових	Так, вдалий дизайн, Якість технічної підтримки	Стратегія лідера. Розширення первинного попиту

Далі треба визначити ринкову позицію проекту в таблиці 4.16.

Таблиця 4.16 - Визначення стратегії позиціонування

	Вимоги до		Ключові	Вибір асоціацій, які
	Швидкість роботи системи і	Стратегія диференц іації	Формування попиту Збільшення	Автоматизація підтримки життєдіяльності

	Простий інтерфейс користувача Детальне налаштовування параметрів для кожного сценарію розвитку Гнучкі ціни Швидкий зв’язок з користувачами		кількості Пристроїв для одного користувача Виявлення нових груп споживачів Нові вектори розвитку існуючої послуги	рослин. Легкий та швидкий старт

4.4 Розробка маркетингової системи стартап-проекту

Щоб розробити маркетингову програму стартап-проєкту потрібно описати маркетингову концепцію товару, визначивши ключові переваги концепції товару(таблиця 4.17). Та визначити межі встановлення ціни, орієнтуючись на ціни товарів-замінників, товарів-аналогів та рівень доходів цільової групи, орієнтуючись на вигоди від його використання(таблиця 4.18).

Таблиця 4.17 - Ключові переваги концепції потенційного товару

№		Вигода, яку	Ключові переваги над
1	Автоматиз ація сценарії підтримки життєдіяльн ості рослин	Економічна ефективність Впевненість у результаті Зручний інтерфейс Низькі ціни	Якість послуги Динамічні сценарії Цінова перевага

Таблиця 4.18 - Межі встановлення ціни

	Ціни	Ціни	Рівень доходів	Верхня та нижня Межі товар/послугу
	5000 – 60000 грн	10000-100000 грн	20000 грн	4000 – 15000 грн

Надалі потрібно визначити оптимальну систему збуту, на основі властивостей продукту (таблиця 4.19). Та сформувати концепції маркетингових комунікацій (таблиця 4.20).

Таблиця 4.19 - Формування системи збуту

	Специфіка поведінки клієнтів	Канали комунікацій, якими цільові клієнти	Ключові позиції, обрані позиціонування	Завдання рекламних повідомлення	Концепція рекламного звернення
	Зрощення рослин вдома або в теплицях	Прямі неофіційні	Послідовна реалізація обраної позиції Доступність та коректність інформації про фірму і товар	Повдомлення цільовій аудиторії Про появу нового продукту	Раціоналістична стратегія

Таблиця 4.20 - Маркетингові комунікації

		Функції збуту,		
	Закупівля здійснюється через тематичні сервіси	Інформування користувачів	Одноріневий канал	Селективна з комбінованим каналом збуту

Три рівні моделі потенційного продукту описано у таблиці 4.21

Таблиця 4.21 – Трирівнева модель продукту

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Продукт за задумом</td><td colspan="3">Розробка системи керування комплексом підтримки життєдіяльності рослин</td></tr>
<tr><td rowspan="2"></td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Кількість елементів
2. Можливість розширення функціоналу
3. Адаптивний шаблон
4. Швидкість побудови і завантаження
5. Швидкість взаємодії</td><td>Нм
М

Нм
М

Нм</td><td>Тх /Тл
Вр/Тх /Тл

Тх /Тл
Тх /Тл

Вр /Тл</td></tr>
<tr><td rowspan="2"></td><td colspan="3">До продажу: масштабування системи, виробництво міні-версій продукту, аналіз сценаріїв розвитку рослин</td></tr>
<tr><td colspan="3">Після продажу: сервіс технічної підтримки</td></tr>
<tr><td colspan="4">В наслідок чого потенційний Продукт буде захищено від копіювання: патенту та офрмленню авторських прав</td></tr>
</table>

ВИСНОВКИ

У даній магістерській дисертації було розглянуто створення системи керування апаратним комплексом підтримки життєдіяльності рослин. Проведено аналіз задачі і існуючих рішень, які обґрунтували актуальність завдання та постановку задачі, задовольнивши усі функціональні і нефункціональні вимоги.

В наслідок детального аналізу вимог було обрано архітектуру для реалізації поточної системи, проведено її декомпозицію на складові модулі та обрано технології, що за вказаними критеріями в межах огляду є оптимальними для реалізації подібного класу систем.

Було створено програмну реалізацію, в основі якої була обрана архітектура, в відповідності до поставленого завдання та реалізовано всі описані задачі, що забезпечило повну відповідність функціональних можливостей системи до тих, що вимагалися з метою досягнення конкурентної здатності програмного засобу. Додавання нових функціональних можливостей було виконано в повному обсязі, що обумовлює актуальність даної роботи. Було проведено мануальне тестування групою користувачів, що оцінили зручність використання застосунку, внесено зміни до зовнішнього вигляду системи відповідно до побажань користувачів, тестування швидкодії додатку, що підвищило зручність користування системою і забезпечило підвищення рівня якості програмного засобу.

Було проведено роботу по аналізу ринкових можливостей запуску стартап-проекту даного програмного засобу, враховано всі ризики, сильні і слабкі сторони часові затрати, стратегії позиціювання та конкурентної поведінки, маркетингові стратегії, та ключові переваги і концепції потенційного продукту, що надало можливість побудови чіткого бізнес-плану подальшого розвитку даної системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alejandro Isabel Luna Maldonado, Julia Mariana Márquez Reyes, Héctor Flores Breceda, Humberto Rodríguez Fuentes, Juan Antonio Vidales Contreras and Urbano Luna Maldonado. Automation and Robotics Used in Hydroponic System Submitted: April 20th 2019Reviewed: November 11th 2019 Published: December 27th 2019 DOI: 10.5772/intechopen.90438;

2. W.Al-Kayssi, A.A.Al-Karaghouli, A.M.HassonS, A.Beker. Influence of soil moisture content on soil temperature and heat storage under greenhouse conditions Volume 45, January–April 1990, Pages 241-252;

3. Dora Neina. The Role of Soil pH in Plant Nutrition and Soil Remediation 03 Nov 2019;

4. Annamaria Castrignano; Agricultural Internet of Things and Decision Support for Precision Smart Farming 2020;

5. NDVI: Normalized Difference Vegetation Index For Agriculture [Електронний ресурс] – Режим доступу: https://eos.com/ndvi/;

6. Ankur Kohli; Smart Plant Monitoring System Using IoT Technology 2020;

7. Сервіс-орієнтована архітектура [Електроний ресурс] – Режим доступу: https://pidruchniki.com/1383121947801/informatika/servis-oriyentovana_arhitektura;

8. The OSI Model Defined, Explained, and Explored [Електронний ресурс] – Режим доступу: https://www.forcepoint.com/cyber-edu/osi-model;

9. UML 2.0 Basic Principles and Background [Електронний ресурс] – Режим доступу: https://sourcemaking.com/uml/basic-principles-and-background/uml2;

10. Що таке Long-Polling, WebSocket, SSE і Comet. [Електронний ресурс]: – Режим доступу: https://myrusakov.ru/long-polling-websocketssse-and-comet.html;

11. What is NoSQL. [Електронний ресурс]: – Режим доступу: https://www.mongodb.com/nosql-explained;

12. Node.js vs. Java [Електронний ресурс]: – Режим доступу: https://www.infoworld.com/article/2883328/nodejs-vs-java-an-epic-battle-for developer-mindshare.html;

13. Mongoose с Node.js - моделирование объектных данных [Електронний ресурс]: – Режим доступу: https://dev-gang.ru/article/mongoose-s-nodejsmodelirovanie-obektnyh-dannyh-409d95da37/;

14. Why Use MongoDB: [Електронний ресурс]: – Режим доступу: https://www.mongodb.com/why-use-mongodb;

15. SPA vs MPA: The definitive guide for decision makers: [Електронний ресурс]: – Режим доступу: https://www.mindk.com/blog/single-page-applications-the-definitive-guide/;

16. 10 Reasons to Consider Angular Framework For Web Development in 2020: [Електронний ресурс]: – Режим доступу: https://www.monocubed.com/angular-framework-for-web-development/;

17. Angular vs React vs Vue: Pros, Cons and When to Use: [Електронний ресурс]: – Режим доступу: https://lanars.com/blog/angular-vs-react-vs-vue;

18. 8 Reasons Why Vue.js is Worth Considering for Your Next Project: [Електронний ресурс]: – Режим доступу: https://www.netguru.com/blog/why-vue-js;

19. Принципы SOLID, о которых должен знать каждый разработчик [Електронний ресурс]: – Режим доступу: https://medium.com/webbdev/solid-4ffc018077da;

20. What the heck is the Virtual DOM? : [Електронний ресурс]: – Режим доступу: https://medium.com/javascript-in-plain-english/what-the-heck-is-the-virtual-dom-3ef1ae4a950b#:~:text=%E2%80%9CThe%20virtual%20DOM%20(VDOM),This%20process%20is%20called%20reconciliation;

21. Real-time communication with WebSocket and Node.js: [Електронний ресурс]: – Режим доступу: https://www.merixstudio.com/blog/real-time-communication-websockets-nodejs/;

22. Понимание HTTP-прокси Nginx, балансировки нагрузки: [Електронний ресурс]: – Режим доступу:https://www.codeflow.site/ru/article/understanding-nginx-http-proxying-load-balancing-buffering-and-caching;

23. JavaScript dependency injection in Node.js : [Електронний ресурс]: – Режим доступу: https://tsh.io/blog/dependency-injection-in-node-js/;

24. Angular Change Detection - How Does It Really Work? : [Електронний ресурс]: – Режим доступу: https://blog.angular-university.io/how-does-angular-2-change-detection-really-work/;

25. How JSON Web Token(JWT) authentication works? : [Електронний ресурс]: – Режим доступу: https://medium.com/@sureshdsk/how-json-web-token-jwt-authentication-works-585c4f076033;

26. Styling An Angular Application With Bootstrap : [Електронний ресурс]: – Режим доступу: https://www.smashingmagazine.com/2019/02/angular-application-bootstrap/;

27. UX/UI For The Modern Web Developer: [Електронний ресурс]: – Режим доступу: https://medium.com/@maloriecasimir/ux-ui-for-web-developers-d5547a4d9b24.